POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Option pricing with Artificial Neural Networks and Wiener-Itô Chaos expansion approximation formulae

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Valentina Buccioni**

Student ID: 969816
Advisor: Prof. Daniele Marazzina
Academic Year: 2021-22

Abstract

This study examines the option pricing method which employs Artificial Neural Networks and Wiener-Itô Chaos asymptotic expansion. In particular, the machine learning model is trained on the difference between the exact option price and its approximation, obtained through the Wiener-Itô Chaos expansion of the underlying. This examination compares this method with the one that applies the Artificial Neural Networks directly on the exact price. Unlike previous research, this study quantifies their prediction accuracy with root mean squared error and mean absolute error measures. In addition to previously examined derivatives, this option pricing procedure is applied to two new ones, i.e. two Down-and-In single barrier options: the first under the Costant Elasticity of Variance and the second under a non linear volatility model. Moreover, the two methods are compared in their best performing scenario, i.e. when the hyper-parameters of the Artificial Neural Networks are tuned. In every case examined, the method that trains on the specified residual term offers more accurate and stable predictions than the one whose Artificial Neural Network trains directly on the option price.

Keywords: Option pricing, Artificial Neural Networks, Wiener-Itô Chaos expansion, Hyper-parameters tuning

Abstract in lingua italiana

Questo studio esamina un metodo di valutazione di opzioni finanziarie che si serve di reti neurali e dello sviluppo in caos di Wiener-Itô. In particolare, il modello di machine learning utilizza la differenza tra il prezzo esatto del derivato e la sua approssimazione, ottenuta attraverso lo sviluppo asintotico in caos di Wiener Itô del sottostante. Questa valutazione mette a confronto questo metodo con quello che applica la rete neurale direttamente al prezzo esatto. Rispetto a studi precedenti, questa trattazione quantifica l'accuratezza delle previsioni attraverso la radice dell'errore quadratico medio e l'errore assoluto medio. Oltre ai derivati già esaminati in precedenza, questa procedura di prezzatura è applicata a due nuovi casi, ovvero due opzioni barriera Down-and-In, il cui sottostante evolve secondo due modelli: uno, a elasticità di variazione costante, l'altro, a volatilità non lineare. Inoltre, i due metodi sono confrontati nel loro scenario migliore, ossia quando gli iperparametri della rete neurale sono ottimizzati. In ogni caso esaminato, il metodo che si serve della differenza tra prezzo esatto e approssimazione offre previsioni più accurate e stabili rispetto a quello la cui rete neurale è applicata direttamente al prezzo esatto.

Parole chiave: Valutazione delle opzioni, Reti Neurali, Sviluppo in caos di Wiener-Itô, Ottimizzazione degli iperparametri

Contents

Introduction

Assessing the fair value of a derivative is a key aspect for option trading. For this reason the theory of option pricing has a long history of contributions which aim to estimate the correct derivative price. Even though Bachelier proposed the first financial model [7], option pricing theory remained dormant until the proposal of the Black-Scholes-Merton model [13] to price European vanilla options. However, this model has the great downside of neglecting the volatility skew and smile shape, so it is not consistent to price options in the market. Many alternatives have been proposed. Usually they fall into the category of local volatility models or stochastic volatility models. The first typology assumes that volatility is a deterministic function of time and the asset price. Some examples are Constant Elasticity of Variance [19] and the Displaced Diffusion model [49]. The second typology assumes that volatility is a random variable. Some examples are the Hull and White [40] and the Heston model [36]. Moreover, there are model variations which combine the characteristics of both. Those are referred as stochastic local volatility models.
Beyond Black & Scholes setting, the model complexity is greater and often there are no closed analytic formulae to calculate the option price. Instead, practitioners adopt numerical techniques such as Monte Carlo and Finite Difference methods. However, they are known to be computationally expensive, especially when the derivative and/or the model of the underlying asset are quite complex.
To overcome this shortcoming, many researchers have focused on the study of approximation formulae of derivative prices. Among these, many adopt an asymptotic expansion approach. Some remarkable examples are Hagan *et al.* [34], and Fouque *et al.* [24], who have studied the singular perturbation technique to obtain European option prices under the SABR and SVMs, respectively. Moreover, Takahashi [62] and Takahashi and Takehara [63] proposed approximated formulae in closed form based on the small volatility asymptotic expansion method for European options when the underlying asset follows SVMs and LVMs.
The asymptotic expansion approximation formulae that are mentioned in this thesis are part of the decennial study of Funahashi and his peers. He has derived closed formulae to approximate the option price where the Wiener-Itô Chaos expansion is applied on the

underlying. In his early contributions [30, 31] he has proposed closed formulae for European derivatives under LVMs and SVMs. Later, Funahashi and Kijima [25] and [58] have extended the method to Asian options. Then, Funahashi and Higuchi [29] and Shiraya [57] have derived approximation formulae for barrier options. In addition, [65] have adopted the Wiener-Itô Chaos expansion approach for American options.
Another solution to ease the computation of derivative prices is the adoption of machine learning techniques. Indeed, in recent years they have become very popular in the financial field to price and calibrate asset pricing models.
Spiegeleer *et al.* [61] proposed the Gaussian process regression, a Bayesian non-parametric technique, as a method to train ANNs to predict American and barrier option prices. This approach allows to obtain prices much faster than conventional pricing methods. However, the offline computation costs for the creation of the training set are quite relevant and the application of this method may be unfeasible in practice.
Other relevant contributions are the one of Horvath *et al.* [39], who have developed a calibration procedure that employs an approximation neural network with implicit training and a calibration layer on top of the network, and the one of Liu *et al.* [46], who have showed that the differential evolution optimizer prevents the calibration model to get stuck on a local minimum. Itkin [41] has described some pitfalls of existing techniques to improve the calibration accuracy and has developed the no-arbitrage condition suitable for the ANNs training. Hernandez [35] has used ANNs to calibrate market data using an inverse mapping method, that speeds-up the procedure compared to alternatives. However, even this promising method has some downsides. Horvath *et al.* [39] have noticed that the inversion function lacks of control, while Itkin [41] has pointed out that the exploding gradient issue occurs when training. Indeed, he explored this concept investigating European Call options under the Black&Scholes setting. Since the option price is an input parameter and the volatility is a target value, during the training is computed the derivative of the volatility with respect to the option price. This quantity is inversely proportional to the Vega, which is a bell-shaped function. This leads to a very high gradient at small and high moneyness and a small gradient in a *at-the-money* scenario. This is referred as the exploding gradient or vanishing problem.
Funahashi [26] combined the application of Artificial Neural Networks and the Wiener-Itô expansion approach to overcome the disadvantages found in previous research. Indeed, the method proposed by Funahashi is based on the employment of the difference between the true price and the asymptotic expansion approximated price as target value. This solves the exploding gradients problem, since the derivative of the residual term with respect to the volatility is no longer bell-shaped, and reduces the computational costs, since accurate results are achieved even with small training sets and simple ANN architectures.

Moreover, Hornik *et al.* [38] have proved that this difference is a smooth and differentiable function: this fact guarantees the application of this method to compute the Greeks.
In [26], Funahashi has compared his proposed method with the one that employs the true price as the ANN label. He provided the results of their application on European Calls and their Delta and Vega under Black&Scholes, on a Up-and-In barrier call option under Heston and on a JPN/USD currency call option under Displaced Diffusion. The results of the proposed method establish that it is accurate and robust. It requires less training data, less layers and less nodes per layer than the method that adopts only the true price as target. This leads to a decrease in the computational costs of offline procedures. Furthermore, it shows that the training behaviour of the Vega in a *at-the-money* scenario is stable, proving that the exploding gradient problem does not occur when applying this method. Moreover, he showed that it correctly reduces the residual term and performs best among its alternatives (i.e. when the approximated price is evaluated with an intrinsic formula, different from the WIC one).
Recently, in [28], Funahashi applied the same method to price options under the SABR model. In this case he employed the difference of the exact implied volatilities and the approximated ones as ANN label value. In this application the training is more robust, predictions are more accurate and the residual term is reduced correctly even in deep *in-the-money* and *out-of-the-money* scenarios, where the approximation closed formula fails to provide precise results.
In this thesis, the work of Funahashi in [26] is investigated in-depth. Firstly, the comparison of the two methods is executed with minor differences to what is done in [26]. The most relevant change resides in the measure choice to assess the predictive accuracy of the models. In [26], Funahashi compares the mean and the variance of the error distributions to have insight on the performances of the methods. In addition to these quantities, this work takes into account the mean absolute error and the root mean squared error. They are metrics used in standard practice to assess the accuracy of ANNs for regression problems.
Then, the two methods have been applied to two options that have been neglected in Funahashi's study: the Down-and-In barrier call under the Constant Elasticity of Variance model and under a non linear volatility model discussed in [21]. In this thesis we answer the question on whether Funahashi's proposed method performs well even in these new cases.
Another novelty proposed in this thesis is the introduction of the hyper-parameters tuning phase. This procedure aims to determine the hyper-parameters for which the validation loss function of the ANN model is minimised [2, 18]. Funahashi does not take into account hyper-parameters such as the learning rate and the batch size, but focuses only on the

ANN architecture. In his research he neglects completely the performance evaluation of the ANN, since he does not adopt a validation set. Moreover, both methods have ANN models with the same set of hyper-parameters. While this type of comparison seems to give equal possibilities to both networks, it may favour the model that performs best with the selected hyper-parameter set. This might lead to erroneous conclusions on the goodness of the methods if one of the ANN models performs much better than the other one [11, 55].

In this thesis the hyper-parameters are tuned so that each model achieves its best performance by minimising its validation loss. Afterwards, the two methods are compared when both networks offer optimal results, producing a fair and more rigorous comparison.

This thesis is organised as follows. In Chapter 1 the financial models used for the underlying and the derivatives priced are described. In Chapter 2 the closed approximation formulae for European Calls and barrier options obtained with the Wiener-Itô Chaos expansion are listed. Additionally, the proof of the closed formulae for the European Call and its numerical results, which have been previously derived in [30, 31], are reported. Chapter 3 deals with a brief explanation of the Artificial Neural Network concepts which are employed in this study. Chapter 4 illustrates the methodology adopted to obtain the fairest comparison of the two methods. Eventually, in Chapter 5 the results are discussed in detail.

1 | Chapter 1: Financial Background

The usual setting to model financial assets and derivatives is considered [26]. Let $(\Omega, \mathscr{F}, \mathbb{P}, \{\mathscr{F}_t\}_{0\leq t\leq T})$ be the filtered probability space, with T finite and the filtration $\{\mathscr{F}_t\}_{0\leq t\leq T}$ which satisfies the usual conditions: it is complete (i.e $\mathscr{F}_0$ contains all $\mathbb{P}$-null sets) and right-continuous (i.e. $\mathscr{F}_t = \mathscr{F}_{t+} := \cap_{s>t}\mathscr{F}_s$). Let $\mathbb{P}$ be the physical measure and $\mathbb{Q}$ the martingale measure. Let $\mathbb{E}$ be the expectation under the $\mathbb{Q}$ martingale measure. In the general setting, the stochastic differential equations of the underlying dynamics and its variance are [26]:

$$\frac{dS_t}{S_t} = r(t)dt + \sigma(S_t, \nu_t; \mathbf{p})dW_t^S \tag{1.1}$$

$$d\nu_t = (\theta(t) - \kappa(t)\nu_t)dt + \gamma(\nu_t)dW_t^\nu. \tag{1.2}$$

By assumption, it has been considered that the parameters r, θ and κ are deterministic functions of time t, that $S_0 > 0$ and that $\nu_0 = \sigma_0^2 > 0$. In addition, dW_t^S and dW_t^ν are $\mathbb{Q}$-Brownian motions such that $dW_t^S dW_t^\nu = \rho dt$. $\sigma(S_t, \nu_t; \mathbf{p})$ is a deterministic function of the asset price S_t, the variance ν and the model parameters vector $\mathbf{p}$.
This thesis deals with five particular cases of the above model. They are the Black and Scholes model [13], the Heston model [36], the Constant Elasticity of Variance model [19] and a non linear volatility model described in [21]. Moreover, the derivatives which are priced are the European vanilla Call, the Up-and-In and the Down-and-In single barrier Call. Additionally, this study focuses on the Greeks, in the specific case of the European vanilla Call under Black & Scholes.

1.1. Black and Scholes

In addition to previous assumptions, this model assumes that the short rate of interest r is constant and that the volatility σ is a strictly positive constant. Therefore, the shapes of the volatility skew and smile are neglected.

The price dynamics of the underlying follows the geometric Brownian motion [13]:

$$\frac{dS_t}{S_t} = rdt + \sigma dW_t. \tag{1.3}$$

1.2. Heston

In addition to the assumptions of the general case (Eqs. (1.1) and (1.2)), this stochastic volatility model (SVM) is characterized by the following volatility equations [26]:

$$\sigma_H(S_t, \nu_t) = \sqrt{\nu_t} \tag{1.4}$$

$$\gamma_H(\nu_t; \epsilon) = \epsilon\sqrt{\nu_t}. \tag{1.5}$$

This results in the following underlying and variance dynamics:

$$\frac{dS_t}{S_t} = rdt + \sigma_H(S_t, \nu_t)dW_t^S \tag{1.6}$$

$$d\nu_t = (\theta - \kappa\nu_t)dt + \gamma_H(\nu_t; \epsilon)dW_t^\nu. \tag{1.7}$$

Further assumptions on the parameters are that r is constant, that θ, κ and ϵ are strictly positive constants and that $\rho \in [-1, 1]$. Moreover, according to the Feller condition, when $2\theta > \epsilon^2$ holds then the process ν is strictly positive [4]. There are no analytic option pricing formulae. Therefore, a Monte Carlo approach will be adopted to evaluate any option price.

1.3. Constant Elasticity of Variance

This local volatility model assumes that its volatility is defined as [19]:

$$\sigma_{CEV}(S_t, t; \mathbf{p}) = \epsilon S_t^{(\beta-1)}, \tag{1.8}$$

where $\mathbf{p} = [\epsilon, \beta]$ is the vector of the model parameters and $\beta \in [0, 1]$ is the elasticity factor. Moreover, given σ as the Black & Scholes volatility, it is assumed that $\epsilon = \epsilon_\beta = \sigma S_0^{(1-\beta)}$. With this assumption, the CEV model represents a generalization of the Black & Scholes model as defined in Eq. (1.3) [47]. Indeed, if $\beta = 1$, then the CEV model is the geometric Brownian motion. Another special case occurs when $\beta = 0$ and the underlying is normally distributed.

In the end, the underlying dynamics is summarised as:

$$\frac{dS_t}{S_t} = rdt + \sigma_{CEV}(S_t, t; \mathbf{p})dW_t^S, \tag{1.9}$$

where r is the constant short interest rate and W_t^S the Wiener process as defined above. Since the CEV model is adopted to price a Down-and-In barrier option, then the computation is done with a Brownian Bridge Monte Carlo approach [33].

1.4. Non linear volatility model

This local volatility model assumes that the volatility is non linear with respect to the asset price S_t. It is defined as [21]:

$$\sigma_{NLV}(S_t, t; \mathbf{p}) = \left(\alpha + \beta\frac{S_t}{S_0}\right) e^{-\mu\frac{S_t}{S_0}}, \tag{1.10}$$

where $\mathbf{p} = [\alpha, \beta, \mu]$ is the vector of the model parameters. α, β are positive constants, while μ is a negative one. Then, the underlying dynamics is described by the equation:

$$\frac{dS_t}{S_t} = rdt + \sigma_{NLV}(S_t, t; \mathbf{p})dW_t^S, \tag{1.11}$$

where r is the constant short interest rate and W_t^S the Wiener process as defined above. Similarly to the CEV model, the computation of the Down-and-In barrier option price is done with the Brownian Bridge Monte Carlo method [33].

1.5. European vanilla Call

A European Call option on a specific underlying asset, with strike K and maturity date T, is a contract written at time $t = 0$ where the holder has the right, but not the obligation, to buy at time $t = T$ the underlying asset at the price K [12]. The term "European" refers to the fact that this contract can be exercised only at expiration date.
The price of the European Call under Black & Scholes is given by the Black & Scholes formula [13]:

$$C(S_t, t) = e^{-r(T-t)}(S_t\mathscr{N}(d_1) - K\mathscr{N}(d_2)), \tag{1.12}$$

where $\mathscr{N}(\cdot)$ is the cumulative distribution function of a standard normal variable, K the option strike, S_t is the spot price at time t, r is the short interest rate, σ is the volatility and d_1 and d_2 are equal to [13]:

$$d_1 = \frac{\log \frac{S_t}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \tag{1.13}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}. \tag{1.14}$$

1.6. Barrier options

Barrier options are financial derivatives whose payoffs depend on the crossing of a certain predefined barrier level (B) by the underlying asset price process (S_t, with $t \in [0, T]$). They are of four types: Down-and-Out, Up-and-Out, Down-and-In and Up-and-In. The first two are knock-out options, i.e. the derivative is active and becomes null if the underlying price process touches or crosses the barrier level. The other two are the knock-in options, i.e. become active as soon as the underlying price process touches or crosses the barrier level. Moreover, the Down typology indicates that the initial spot price S_0 is assumed to be above the barrier level B. Conversely, the Up type assumes that it is below.

In this thesis we consider only the knock-in derivatives. Specifically, we evaluate a Up-and-In and a Down-and-In Call, in European exercise style.

Assuming the underlying is based on the geometric Brownian motion, closed formulae for barrier options pricing are the ones stated by Rubinstein *et al.* [53] or, alternatively, through the static replication of a portfolio of vanilla options, a method pioneered by Carr [16].

However, when the underlying does not follow a geometric Brownian motion, such as in this study, a viable approach is Monte Carlo or the Brownian-Bridge Monte Carlo [33], which is specific for path-dependent options.

1.7. Greeks

The Greeks represent the sensitivity measures of the portfolio of derivatives based on a single underlying. These measures are useful for two main reasons: they quantify the risk exposure of the portfolio due to the changes in the underlying process; they reflect the sensitivity of the portfolio to misspecifications of the model parameters [12].

However, in this particular case where the portfolio consists of a single derivative, the Greeks indicate the changes of the option price with respect to the variations of the underlying price or the model parameters. Given $C(t, s)$ the option price at time t based on the underlying price process S_t [12]:

$$\Delta = \frac{\partial C(t,s)}{\partial s}, \tag{1.15}$$

$$\Gamma = \frac{\partial^2 C(t,s)}{\partial s^2}, \tag{1.16}$$

$$\rho = \frac{\partial C(t,s)}{\partial r}, \tag{1.17}$$

$$\Theta = \frac{\partial C(t,s)}{\partial t} \tag{1.18}$$

$$\upsilon = \frac{\partial C(t,s)}{\partial \sigma}. \tag{1.19}$$

Moreover, for the European Call under Black & Scholes scenario there are analytic formulae for its Greeks. In this thesis, only Delta and Vega are taken into account: here are reported the formulae only for those two sensitivity measures [13]:

$$\Delta(t) = \mathscr{N}(d_1) \tag{1.20}$$

$$\upsilon(t) = \psi(d_1) S_t \sqrt{T-t}, \tag{1.21}$$

where $\psi(d_1)$ is the probability density function of a standard normal evaluated in d_1 (Eq.(1.13)). In particular, Delta and Vega indicate changes of the derivative price with respect to the variations of the underlying price and of the volatility parameter, respectively.

2 | Chapter 2: Asymptotic Expansion

In this thesis, the asymptotic expansion described in detail by Funahashi *et al.* in [29–32] has been considered. This chapter is structured as follows: first, the proof of the European Call asymptotic formula is developed in depth; second, some numerical examples are presented, which verify the accuracy of the asymptotic formula; third, the asymptotic formulae for single barrier options are stated.

2.1. Derivation of the EU Call asymptotic formula

In brief, the Wiener-Itô chaos (WIC) expansion procedure adopted to obtain the asymptotic formula to price European Call options is the following:

1. writing the underlying dynamics using Hermite polynomials;
2. building the expansion of the underlying dynamics through successive substitutions;
3. approximating this expansion by a truncated sum of Itô integrals - in this case until the third order of the WIC expansion;
4. deriving the probability density function of the approximated underlying variable;
5. elaborating the closed formula of the financial option on the underlying considered, by means of its probability density function.

It assumed that the underlying asset follows the dynamics given by the SDE:

$$\frac{dS_t}{S_t} = r(t)dt + \sigma(S_t, t)dW_t, \tag{2.1}$$

where the short rate of interest $r(t)$ is a deterministic function of time, the volatility $\sigma(s,t)$ is a deterministic function of the asset price and time and $\{W_t\}_{t\geq 0}$ is a standard Brownian motion under $\mathbb{Q}$ risk-neutral measure. Despite the fact that Eq.(2.1) indicates a simpler

dynamics than the one in Eq.(1.1), there is no loss of generality in the following proof. However, in Lemma 3 and the Appendices A and B, the formulae are stated assuming the most general dynamics of Eq.(1.1) for completeness.
Applying Itô's formula and substituting the forward price $F(0,t) = S_0 e^{\int_0^t r(s)\,ds}$, we obtain:

$$\begin{aligned} S_t &= S_0 \exp\left[\int_0^t (r(u) - \frac{1}{2}\sigma^2(S_u,u))\,du + \int_0^t \sigma(S_u,u)\,dW_u\right] \\ &= F(0,t)\exp\left[\int_0^t \sigma(S_u,u)\,dW_u - \int_0^t \frac{1}{2}\sigma^2(S_u,u)\,du\right] \\ &= F(0,t)\exp\left[J_t(\sigma) - \frac{1}{2}\|\sigma\|_t^2\right], \end{aligned} \tag{2.2}$$

where $J_t(g) = \int_0^t g(u)\,dW_u$ and $\|g\|_t^2 = \int_0^t g^2(u)\,du$.
Let's consider the definition of probabilists' Hermite polynomials [3]:

$$h_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{-\frac{x^2}{2}}, n = 1,2,3... \tag{2.3}$$

with $h_0 = 1$.
Given this definition, it is possible to state this useful Hermite expansion Lemma, which detailed proof is at page 16 of Di Nunno *et al.* [20]:

Lemma 1. *For any $x \in \mathbf{R}$ and $\lambda > 0$, we have:*

$$\exp\left[tx - \frac{(t\sqrt{\lambda})^2}{2}\right] = \sum_{n=0}^{\infty} \frac{(t\sqrt{\lambda})^n}{n!} h_n\left(\frac{x}{\sqrt{\lambda}}\right). \tag{2.4}$$

The expansion (2.4) is valid for a random variable too. Indeed, according to [20], imposing $\sigma \in L^2([0,T])$, $t = 1$, $x = J_t(\sigma)$ and $\lambda = \|\sigma\|_t^2$:

$$\exp\left[J_t(\sigma) - \frac{1}{2}\|\sigma\|_t^2\right] = \sum_{n=0}^{\infty} \frac{\|\sigma\|_t^n}{n!} h_n\left(\frac{J_t(\sigma)}{\|\sigma\|_t}\right). \tag{2.5}$$

In [31] it is observed that the truncation of the right hand side of Eq.(2.5) always provides a better approximation of its left hand side than, for example, the Maclaurin expansion truncation. The former has lower mean squared errors than the latter for the volatility levels observed in the market. Moreover, its superiority increases as the volatility and the maturity are higher. This justifies the writing of the expression in Eq.(2.2) using Hermite polynomials expansion.

Furthermore, according to (1.14) in [67] and in [42], it is known that:

$$\frac{\|\sigma\|_t^n}{n!} h_n\left(\frac{J_t(\sigma)}{\|\sigma\|_t}\right) = \int_0^t \int_0^{t_n} \dots \int_0^{t_2} \sigma(t_1)\sigma(t_2)\dots\sigma(t_n)\, dW_{t_1} \dots dW_{t_n}, \tag{2.6}$$

where $\{W_t\}_{t\geq 0}$ is a one-dimensional Brownian motion under $\mathbb{Q}$ risk-neutral measure. An additional useful property is that the right hand side of Eq.(2.6) converges rapidly to 0 as n increases under specific conditions. This is stated in Proposition 2.1 below, whose proof is in Appendix B of [31]:

Proposition 2.1. *Consider the iterated integral*

$$I_n = \int_0^t \int_0^{t_n} \dots \int_0^{t_2} \sigma_1(t_1)\sigma_2(t_2)\dots\sigma_n(t_n)\, dW_{t_1} \dots dW_{t_n}.$$

If the volatilities $\sigma_k(t)$ are deterministic functions and $\overline{\sigma}(t) = max_k \sigma_k(t) \in L^2([0,t])\ \forall t$, then we have $\mathbb{E}[I_n^2] \leq \frac{\|\overline{\sigma}\|_t^{2n}}{n!}$.

For this reason, if $\|\overline{\sigma}\|$ is small enough, then the sum of iterated integrals of order greater than n can be considered zero in the L^2 sense for large n. Thus, rewriting Eq.(2.2) as:

$$\begin{aligned}\frac{S_t}{F(0,t)} &= \exp\left[J_t(\sigma) - \frac{1}{2}\|\sigma\|_t^2\right] \\ &= 1 + \sum_{n=1}^{\infty} \int_0^t \int_0^{t_n} \dots \int_0^{t_2} \sigma(t_1)\sigma(t_2)\dots\sigma(t_n)\, dW_{t_1} \dots dW_{t_n} \end{aligned}\tag{2.7}$$

and applying Proposition 2.1, the iterated integrals higher than the third order are neglected in this proof.

Before proceeding with the proof, we notice that under some particular conditions, the stochastic process $\{X_t; 0 \leq t \leq T\}$ that satisfies the stochastic integral equation

$$X_t = X_0 \exp\left[\int_0^t a(s)\, ds - \frac{1}{2}\int_0^t b^2(X_s, s)\, ds + \int_0^t b(X_s, s)\, dW_s\right] \tag{2.8}$$

can be constructed by successive substitution. Indeed, given $X_t^{(0)} = X_0 e^{\int_0^t a(s)\, ds}$ and

$$X_t^{(k+1)} = X_0 \exp\left[\int_0^t a(s)\, ds - \frac{1}{2}\int_0^t b^2(X_s^{(k)}, s)\, ds + \int_0^t b(X_s^{(k)}, s)\, dW_s\right], \tag{2.9}$$

with $a(s)$ and $b(X_s^{(k)}, s)$ that follow some regularity conditions, then $X_t^{(k)}$ converges almost surely to the solution X_t.
A sufficient condition on $a(s)$ and $b(X_s^{(k)}, s)$ is given for the convergence and it is stated below for completeness. Its proof is omitted in this thesis, since it is similar to the proof of the existence of the strong solution of the Eq. (2.1). See [66] for the detailed proof.

Proposition 2.2. *Let T>0 and suppose that a(t) and b(t,x) satisfy*

$$|b(t,x)^2| + |b(t,x)| \le C(1 + |log(x)|), \;\; x \in \mathbb{R}, \;\; t \in [0,T],$$
$$|b(t,x)^2 - b(t,y)^2| + |b(t,x) - b(t,y)| \le D\left|log\left(\frac{x}{y}\right)\right|, \;\; x \in \mathbb{R}, \;\; t \in [0,T],$$

for some constant C and D. Then $X_t^{(k)}$ converges to the solution X_t almost surely.

As stated in [31], this condition is usually too strong for practical purposes. Therefore, in this thesis it is assumed that successive substitution produces the stochastic integral equation solution without checking the above condition.
Thus, the successive substitution construction of the solution can be applied to this specific case too. The solution of Eq.(2.1), that is:

$$S_t = F(0,t)\exp\left[\int_0^t \sigma(S_u, u)\, dW_u - \frac{1}{2}\int_0^t \sigma^2(S_u, u)\, du\right], \tag{2.10}$$

can be constructed by successive substitution. Let $S_t^{(0)} = F(0,t)$ and:

$$S_t^{(m+1)} = F(0,t)\exp\left[J_t(\sigma_m) - \frac{1}{2}||\sigma_m||_t^2\right], \tag{2.11}$$

where $\sigma_m = \sigma(S_t^{(m)}, t)$. By assumption, $S_t^{(m)}$ converges to S_t as $m \to \infty$. Hence:

$$S_t = S_t^{(1)} + \sum_{m=1}^{\infty}\{S_t^{(m+1)} - S_t^{(m)}\}. \tag{2.12}$$

Using Eqs. (2.11) and (2.5), it is possible to express $S_t^{(m)}$ with the Hermite polynomials expansion:

$$\frac{S_t^{(m+1)}}{F(0,t)} = 1 + \sum_{n=1}^{\infty}\frac{||\sigma_m||_t^n}{n!}h_n\left(\frac{J_t(\sigma_m)}{||\sigma_m||_t}\right) \tag{2.13}$$

and, thus, define:

$$I_{m,n}(t) = \frac{1}{n!}\left[\frac{\|\sigma_m\|_t^n}{n!}h_n\left(\frac{J_t(\sigma_m)}{\|\sigma_m\|_t}\right) - \frac{\|\sigma_{m-1}\|_t^n}{n!}h_n\left(\frac{J_t(\sigma_{m-1})}{\|\sigma_{m-1}\|_t}\right)\right]. \tag{2.14}$$

Therefore, Eq.(2.12) can be written as:

$$S_t = S_t^{(1)} + F(0,t)\sum_{m,n=1}^{\infty} I_{m,n}(t). \tag{2.15}$$

The objective is to truncate the sum of iterated integrals at the third order, *i.e.* for $m+n \leq 3$. First, the term $S_t^{(1)}$ can be approximated as:

$$\begin{aligned}\widetilde{S}_t^{(1)} = F(0,t)\Bigg[1 &+ \int_0^t \sigma_0(t_1)\,dW_{t_1} + \int_0^t\int_0^{t_2} \sigma_0(t_1)\sigma_0(t_2)\,dW_{t_1}\,dW_{t_2} \\ &+ \int_0^t\int_0^{t_3}\int_0^{t_2} \sigma_0(t_1)\sigma_0(t_2)\sigma_0(t_3)\,dW_{t_1}\,dW_{t_2}\,dW_{t_3}\Bigg].\end{aligned} \tag{2.16}$$

Indeed, $\sigma_0(t) = \sigma(F(0,t),t)$ is a deterministic function, so Eq.(2.7) can be applied and its sum can be truncated to the third iterated integral, resulting in Eq.(2.16).

Next, the terms $I_{m,n}(t)$ are approximated with an iterated integral with deterministic volatilities. To accomplish this, Taylor's expansion around $S_t^{(m)}$ of $J_t(\sigma_m)$ is employed. Recalling that $J_t(\sigma_m) = \int_0^t \sigma_m(u)\,dW_u = \int_0^t \sigma(S_u^{(m)},u)\,dW_u$, it results in:

$$\begin{aligned}J_t(\sigma_m) \approx J_t(\sigma_{m-1}) &+ \int_0^t \sigma'_{m-1}(u)\{S_u^{(m)} - S_u^{(m-1)}\}\,dW_u \\ &+ \frac{1}{2}\int_0^t \sigma''_{m-1}(u)\{S_u^{(m)} - S_u^{(m-1)}\}^2\,dW_u\end{aligned} \tag{2.17}$$

and

$$J_t^2(\sigma_m) \approx J_t^2(\sigma_{m-1}) + 2J_t(\sigma_{m-1})\int_0^t \sigma'_{m-1}(u)\{S_u^{(m)} - S_u^{(m-1)}\}\,dW_u, \tag{2.18}$$

where $\sigma'_m(t) = \partial_x\sigma(x,t)|_{x=S_t^{(m)}}$, and $\sigma''_m(t) = \partial_{xx}\sigma(x,t)|_{x=S_t^{(m)}}$. The following Lemma states the iterated integrals required up to the third order. The detailed proof is in [31].

Lemma 2. *Each iterated integral $I_{m,n}(t)$ defined in Eq.(2.14) is approximated as follows:*

$$\begin{aligned} I_{1,1}(t) \approx & \int_0^t \sigma_0^{'}(s)F(0,s)\left(\int_0^s \sigma_0(u)\,dW_u\right)dW_s \\ &+ \int_0^t \sigma_0^{'}(s)F(0,s)\left(\int_0^s \sigma_0(u)\left(\int_0^u \sigma_0(r)\,dW_r\right)dW_u\right)dW_s \\ &+ \int_0^t \sigma_0^{''}(s)F^2(0,s)\left(\int_0^s \sigma_0(u)\left(\int_0^u \sigma_0(r)\,dW_r\right)dW_u\right)dW_s \\ &+ \frac{1}{2}\int_0^t \sigma_0^{''}(s)F^2(0,s)\left(\int_0^s \sigma_0^2(u)\,dW_u\right)dW_s, \end{aligned} \tag{2.19}$$

$$\begin{aligned} I_{1,2}(t) \approx & \int_0^s \sigma_0(s)\left(\int_0^s \sigma_0^{'}(u)F(0,u)\left(\int_0^u \sigma_0(r)\,dW_r\right)dW_u\right)dW_s \\ &+ 2\int_0^t \sigma_0^{'}(s)F(0,s)\left(\int_0^s \sigma_0(u)\left(\int_0^u \sigma_0(r)\,dW_r\right)dW_u\right)dW_s \\ &+ \int_0^t \sigma_0^{'}(s)F(0,s)\left(\int_0^s \sigma_0^2(u)\,dW_u\right)dW_s, \end{aligned} \tag{2.20}$$

$$I_{2,1}(t) \approx \int_0^t \sigma_0^{'}(s)F(0,s)\left(\int_0^s \sigma_0^{'}(u)F(0,u)\left(\int_0^u \sigma_0(r)\,dW_r\right)dW_u\right)dW_s. \tag{2.21}$$

For $m+n \geq 4$, we have $I_{m,n}(t) \approx 0$.

At last, the approximated formula is obtained by substituting the terms in Eq.(2.15) with their respective approximations: $\widetilde{S}_t^{(1)}$, $I_{1,1}(t)$, $I_{1,2}(t)$ and $I_{2,1}(t)$. The resulting third order WIC expansion obtained is stated in the following Lemma 3 [26]:

Lemma 3. *Let us denote $\sigma^{(0)}(t) = \sigma(F(0,t), V(0,t))$ and $\gamma^{(0)}(t) = \gamma(V(0,t))$ where $F(0,t) = e^{\int_0^t r(s)\,ds}$ and $V(0,t) = \overline{E}(t)(\nu_0 + \int_0^t E(u)\theta(u)\,du)$ with $E(t) = e^{\int_0^t \kappa(u)\,du}$ and $\overline{E}(t) = 1/E(t)$, the asset price in section 1.1 is approximated by:*

$$S_t = F(0,t)(1 + a_1(t) + a_2(t) + a_3(t) + R_4), \tag{2.22}$$

where $F(0,t) = e^{\int_0^t r(s)\,ds}$ and

$$a_1(t) = \int_0^t p_1(s)\,dW_s^S, \tag{2.23}$$

$$a_2(t) = \int_0^t p_2(s)\left(\int_0^s \sigma^{(0)}(u)\, dW_u^S\right) dW_s^S + \int_0^t p_3(s)\left(\int_0^s p_4(u)\, dW_u^\nu\right) dW_s^S, \tag{2.24}$$

$$\begin{aligned}
a_3(t) = &\int_0^t p_5(s)\left(\int_0^s \sigma^{(0)}(u)\left(\int_0^u \sigma^{(0)}(r)\, dW_r^S\right) dW_u^S\right) dW_s^S \\
+ &\int_0^t p_6(s)\left(\int_0^s p_4(u)\left(\int_0^u p_4(r)\, dW_r^\nu\right) dW_u^\nu\right) dW_s^S \\
+ &\int_0^t p_7(s)\left(\int_0^s p_4(u)\left(\int_0^u \sigma^{(0)}(r)\, dW_r^S\right) dW_u^\nu\right) dW_s^S \\
+ &\int_0^t p_7(s)\left(\int_0^s \sigma^{(0)}(u)\left(\int_0^u p_4(r)\, dW_r^\nu\right) dW_u^S\right) dW_s^S \\
+ &\int_0^t p_2(s)\left(\int_0^s p_8(u)\left(\int_0^u \sigma^{(0)}(r)\, dW_r^S\right) dW_u^S\right) dW_s^S \\
+ &\int_0^t p_2(s)\left(\int_0^s p_3(u)\left(\int_0^u p_4(r)\, dW_r^\nu\right) dW_u^S\right) dW_s^S \\
+ &\int_0^t p_3(s)\left(\int_0^s \gamma_\nu^{(0)}(u)\left(\int_0^u p_4(r)\, dW_r^\nu\right) dW_u^\nu\right) dW_s^S,
\end{aligned} \tag{2.25}$$

where $\gamma_\nu^{(0)}(t) = \partial_\nu \gamma(\nu)\big|_{\nu=\nu_t^{(0)}}$*, and deterministic functions* Σ_t *and* $p_k(t)$ *being defined in Appendix A.*

Two important clarifications of the above Lemma 3 are needed. First, R_n indicates the *n-1-* order truncation remainder: in the case of Eq.(2.22), it contains all multiple stochastic integrals after the sum truncation of the third order. According to Proposition 2.1, the remainder R_n is null in L^2 sense for large n [32]. This fact is recalled since it is truly relevant for this thesis. Indeed, as an ANN is built to predict this remainder term, then the option price is evaluated with the asymptotic formula: without this proven fact, its convergence to the exact price for large n would not be assured.
Second, the above Lemma 3 states the approximated formula for the more general underlying dynamics given by Eq.(1.1) and not for the more specific case of Eq.(2.1). Clearly, the approximated formula for the dynamics of Eq.(2.1) can be retrieved trivially.

The aim of the following steps is to obtain the probability distribution of S_t. Without loss of generality, it is defined:

$$X_t := \frac{S_t}{F(0,t)} - 1 \approx a_1(t) + a_2(t) + a_3(t)$$

It is observed that $a_1(t)$ follows a normal distribution centred in zero and with variance $\Sigma_t := \int_0^t p_1^2(s)\,ds$. The moments of $a_j(t)$ conditional to $a_1(t)$ can be obtained explicitly. This and the following steps are based on the work of Takahashi [62] and Yoshida [64]. Taking into account the characteristic function of X_t, denoted as $\Psi(\xi) := \mathbb{E}\left[e^{i\xi X_t}\right]$, it can be approximated as:

$$\begin{aligned}\Psi(\xi) &\approx \mathbb{E}\left[e^{i\xi(a_1(t)+a_2(t)+a_3(t)}\right]\\ &= \mathbb{E}\left[e^{i\xi(a_1(t)}(1+i\xi a_2(t)+i\xi a_3(t)-\frac{1}{2}\xi^2 a_2^2(t)+R_4)\right].\end{aligned}$$

Recalling Proposition 2.1, the remainder R_4 can be neglected. Indeed:

$$\begin{aligned}|\mathbb{E}\left[e^{i\xi a_1(t)}R_4\right]| &\leq \mathbb{E}\left[|e^{i\xi a_1(t)}R_4|\right]\\ &\leq (\mathbb{E}[|e^{i\xi a_1(t)}|^2])^{\frac{1}{2}}(\mathbb{E}[|R_4|^2])^{\frac{1}{2}}\\ &= (\mathbb{E}[|R_4|^2])^{\frac{1}{2}} \approx 0.\end{aligned}$$

Evaluating the conditional expectation with respect to $a_1(t)$:

$$\begin{aligned}\Psi(\xi) \approx \mathbb{E}\left[e^{i\xi a_1(t)}\right] &+ i\xi\mathbb{E}\left[e^{i\xi a_1(t)}\mathbb{E}\left[a_2(t)|a_1(t)\right]\right]\\ &+ i\xi\mathbb{E}\left[e^{i\xi a_1(t)}\mathbb{E}\left[a_3(t)|a_1(t)\right]\right]-\frac{1}{2}\xi^2\mathbb{E}\left[e^{i\xi a_1(t)}\mathbb{E}\left[a_2^2(t)|a_1(t)\right]\right].\end{aligned} \tag{2.26}$$

The conditional expectations can be evaluated explicitly, according to Lemma 2.1 in Takahashi's work [62], are the following:

$$\mathbb{E}\left[a_2(t)|a_1(t)=x\right] = q_1(t)\left(\frac{x^2}{\Sigma_t^2}-\frac{1}{\Sigma_t}\right), \tag{2.27}$$

$$\mathbb{E}\left[a_3(t)|a_1(t)=x\right] = q_2(t)\left(\frac{x^3}{\Sigma_t^3}-\frac{3x}{\Sigma_t^2}\right), \tag{2.28}$$

$$\mathbb{E}\left[a_2^2(t)|a_1(t)=x\right] = q_3(t)\left(\frac{x^4}{\Sigma_t^4}-\frac{6x^2}{\Sigma_t^3}+\frac{3}{\Sigma_t^2}\right)+q_4(t)\left(\frac{x^2}{\Sigma_t^2}-\frac{1}{\Sigma_t}\right)+q_5(t), \tag{2.29}$$

where Σ_t and $q_k(t)$ are defined in Appendix A.
Next, the characteristic function can be inverted to retrieve the density function of X_t, according to the following Lemma 4 [31], whose proof can be found in Takahashi's work [62]:

Lemma 4. *Suppose that X follows the normal distribution with zero mean and variance Σ. Then, for any polynomial functions $f(x)$ and $g(x)$, we have*

$$\frac{1}{2\pi}\int_{\mathbf{R}} e^{-iky} g(-ik)\mathbb{E}\left[f(X)e^{ikX}\right] dk = g\left(\frac{\partial}{\partial y}\right) f(y)n(y;0,\Sigma),$$

where n(x;a,b) denotes the normal density function with mean a and variance b.

Applying the above Lemma 4 to Eq.(2.26), the density function f_{X_t} is obtained as follows:

$$\begin{aligned} f_{X_t}(x) = n(x;0,\Sigma_t) &- \frac{\partial}{\partial x}\{\mathbb{E}\left[a_2(t)|a_1(t)=x\right] n(x;0,\Sigma_t)\} \\ &- \frac{\partial}{\partial x}\{\mathbb{E}\left[a_3(t)|a_1(t)=x\right] n(x;0,\Sigma_t)\} \\ &+ \frac{1}{2}\frac{\partial^2}{\partial x^2}\{\mathbb{E}\left[a_2^2(t)|a_1(t)=x\right] n(x;0,\Sigma_t)\}. \end{aligned} \tag{2.30}$$

By substituting the Eqs. (2.27), (2.28) and (2.29) into Eq.(2.30), it follows [31]:

Theorem 2.1. *The probability density function of X_t is approximated as:*

$$\begin{aligned} f_{X_t}(x) \approx \widetilde{f}_{X_t}(x) = \frac{1}{2\Sigma_t^6} n(x;0,\Sigma_t)\Big[&q_3(t)(x^6 - 15x^4\Sigma_t + 45x^2\Sigma_t^2 - 15\Sigma_t^3) \\ &+ \Sigma_t^2(2q_2(t)+q_4(t))(x^4 - 6x^2\Sigma_t + 3\Sigma_t^2) \\ &+ \Sigma_t^3\left[2q_1(t)(x^3 - 3x\Sigma_t) + q_5(t)(x^2\Sigma_t - \Sigma_t^2) + 2\Sigma_t^3\right]\Big], \end{aligned} \tag{2.31}$$

where n(x;a,b) denotes the normal density function with mean a and variance b.

An alternative expression of the density function $\widetilde{f}_{X_t}$ employs Hermite polynomials, defined earlier in Eq.(2.3), is the following:

$$\begin{aligned} \widetilde{f}_{X_t}(x) = \frac{1}{2} n(x;0,\Sigma_t)\Bigg[&\frac{q_3(t)}{\Sigma_t^3} h_6\left(\frac{x}{\sqrt{\Sigma_t}}\right) + \frac{(2q_2(t)+q_4(t))}{\Sigma_t^2} h_4\left(\frac{x}{\sqrt{\Sigma_t}}\right) \\ &+ \frac{2q_1(t)}{(\sqrt{\Sigma_t})^3} h_3\left(\frac{x}{\sqrt{\Sigma_t}}\right) + \frac{q_5(t)}{\Sigma_t} h_2\left(\frac{x}{\sqrt{\Sigma_t}}\right) + 2\Bigg]. \end{aligned}$$

To retrieve the approximated density function in terms of the underlying asset, this formula is applied:

$$\widetilde{f}_{S_t}(x) = \frac{\widetilde{f}_{X_t}\left(\frac{x}{F(0,t)} - 1\right)}{F(0,t)}. \tag{2.32}$$

Remark 1. *The integral of the approximated density $\widetilde{f}_{X_t}$ over the entire space might not be unity. In that case, it can be replaced by $\hat{f}_{X_t} := \frac{\widetilde{f}_{X_t}(x)}{\int_{\mathbf{R}} \widetilde{f}_{X_t}(x)\,dx}$, so that $\int_{\mathbf{R}} \hat{f}_{X_t}(x)\,dx = 1$ [31].*

Eventually, it is possible to evaluate trivially the European Call option pricing formula, given strike K and maturity T.

$$C(T) = \mathbb{E}\left[e^{-\int_0^T r(s)\,ds}(S_T - K)^+\right] = F(0,T)\mathbb{E}\left[e^{-\int_0^T r(s)\,ds}(X_T + \widetilde{K})^+\right]$$

where $\widetilde{K} = 1 - \frac{K}{F(0,t)}$. Then, given the density function f_{X_t}, the European call option price is the result of the following integral:

$$C(T) = S(0)\int_{-\widetilde{K}}^{\infty}(x + \widetilde{K})f_{X_t}(x)\,dx,$$

and it is stated in the following theorem [26]:

Theorem 2.2. *The European call option price of S with maturity T and strike K is approximated by that of $\overline{S} = F(0,t)(1 + a_1(t) + a_2(t) + a_3(t))$*

$$\begin{aligned} C^{\overline{S}}(T,K) = \frac{S_0 n(\widetilde{K};0,\Sigma_T)}{2\Sigma_T^4}&[q_3(T)(\widetilde{K}^4 - 6\widetilde{K}^2\Sigma_T + 3\Sigma_T^2) \\ &+ \Sigma_T^2(q_4(T) + 2q_2(T))(\widetilde{K}^2 - \Sigma_T) \\ &+ \Sigma_T^3(-2q_1(T)\widetilde{K} + q_5(T)\Sigma_T + 2\Sigma_T^2)] \\ &+ \frac{S_0\widetilde{K}}{\sqrt{2}}\left(1 - \Phi\left(\frac{-\widetilde{K}}{\sqrt{\Sigma_T}}\right)\right), \end{aligned} \tag{2.33}$$

where $\widetilde{K} := 1 - \frac{K}{F(0,t)}$ and $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. The deterministic functions $q_i(t)$ and Σ_t are defined in Appendix A.

2.2. Numerical examples

In these numerical examples presented, we assume an underlying dynamic such as the one in Eq.(2.1) and a volatility term such that $\sigma(S,t) := \nu(t)S^{\beta(t)-1}$, where $\nu(t)$ and $\beta(t)$ are deterministic functions. In particular, let $\nu(t) = \nu$ and $\beta(t) = 1$, so that the underlying has a geometric Brownian motion dynamic as in the Black and Scholes model [13].
Two tests are executed. They replicate the numerical examples in [30] and in [31], re-

spectively. First, it is checked whether the density function $\widetilde{f}_{S_t}$ is a good approximation of f_{S_t} or not. Second, the European Call prices are computed with both the Black and Scholes closed formula and the approximated one in Eq.(2.33) and compared in low and high volatility scenarios.

In the specific case of the Black and Scholes model, S_t is distributed as a log-normal with mean $(r - \frac{\nu^2}{2})T + ln(S_0)$ and variance $\nu\sqrt{T}$, where S_0 indicates the initial underlying price at time 0 and T the maturity in years. In the following figures, it is assumed that $\nu = 0.15$, $S_0 = 80$, $r = 0.03$ and $T = 5$ years. "Analytic" indicates the probability density function evaluated as a log-normal, "WIC(3)" indicates the one obtained with Eqs. (2.31) and (2.32), "WIC(2)" and "WIC(1)" indicate the densities derived from the truncation, respectively, at the second and first order. Figure 2.1 shows that the WIC(3) approximation is accurately close to the analytic density.

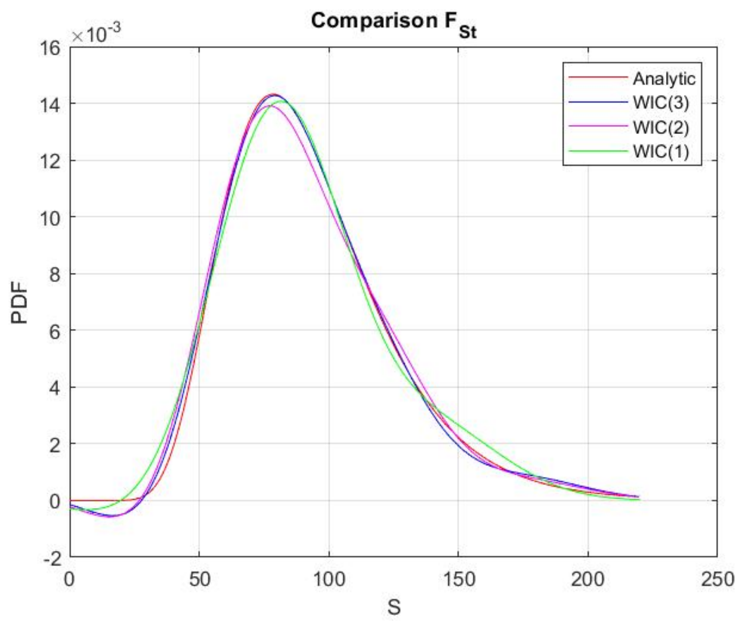


Figure 2.1: Comparison of the exact density of the underlying S_t and its Wiener-Itô Chaos approximations, whose expansions are truncated at the third (WIC(3)), second (WIC(2)) and first (WIC(1)) order.

Strike	**BS**	**WIC3**	**RE [%]**
T = 3 months			
84.735	0.92931	0.92932	0.001823
82.643	1.5441	1.5441	0.00040321
80.602	2.3931	2.3931	-4.9442e-06
78.612	3.4812	3.4812	-0.00018286
76.671	4.7856	4.7856	-0.00033005
T = 1 year			
91.106	1.9034	1.9037	0.014785
86.663	3.124	3.1241	0.0031514
82.436	4.7828	4.7828	-7.9123e-05
78.416	6.8733	6.8732	-0.0014974
74.592	9.3353	9.3351	-0.0026202
T = 5 years			
116.24	4.4993	4.5071	0.17273
103.94	7.1679	7.1702	0.033286
92.947	10.655	10.655	-0.0019802
83.115	14.872	14.869	-0.017718
74.323	19.627	19.622	-0.028928
T = 10 years			
148.15	6.6128	6.6465	0.50972
126.49	10.308	10.317	0.09029
107.99	14.998	14.997	-0.0079312
92.196	20.5	20.49	-0.052246
78.712	26.509	26.487	-0.081513

Table 2.1: Evaluation of EU Call prices with Black&Scholes and the Wiener-Itô chaos third order expansion, with varying strike and maturity T. The relative error defined in Eq.(2.34) shows in percentage the accuracy of the approximated prices. This table shows the scenario of low volatility ($\nu = 0.15$).

Strike	**BS**	**WIC3**	**RE [%]**
T = 3 months			
84.735	3.124	3.1241	0.0031514
82.643	3.8977	3.8978	0.0012974
80.602	4.7828	4.7828	-7.9123e-05
78.612	5.7767	5.7767	-0.00099268
76.671	6.8733	6.8732	-0.0014974
T = 1 year			
91.106	6.3816	6.3831	0.024076
86.663	7.867	7.8678	0.0095716
82.436	9.5388	9.5387	-0.001267
78.416	11.385	11.384	-0.008514
74.592	13.387	13.386	-0.012543
T = 5 years			
116.24	14.872	14.907	0.24099
103.94	17.823	17.838	0.085623
92.947	21.015	21.008	-0.031808
83.115	24.4	24.373	-0.11159
74.323	27.921	27.878	-0.15655
T = 10 years			
148.15	21.476	21.612	0.62944
126.49	25.233	25.283	0.19882
107.99	29.179	29.142	-0.12788
92.196	33.242	33.125	-0.35199
78.712	37.343	37.164	-0.47928

Table 2.2: Evaluation of EU Call prices with Black&Scholes and the Wiener-Itô chaos third order expansion, with varying strike and maturity T. The relative error defined in Eq.(2.34) shows in percentage the accuracy of the approximated prices. This table shows the scenario of high volatility ($\nu = 0.3$).

The numerical experiments of the European Call prices consider two volatility scenarios, with $\nu = 0.15$ and $\nu = 0.3$. Other parameters are set such as: $S_0 = 80$, $r = 0.03$,

$T = 0.25, 1, 5, 10$ years and the strikes are given by $K_i(T) = F(0,T)e^{(0.1\sqrt{T}\delta_i)}$, with $\delta_i = -1.0, -0.5, 0, 0.5, 1.0$ and $i = 1, 2, 3, 4, 5$. To check the accuracy, the relative error (RE) is evaluated as follows:

$$RE = \frac{Approximated\ value - Exact\ value}{Exact\ value}, \tag{2.34}$$

and it is reported in Tables 2.1 and 2.2 as a percentage. The accuracy of the approximated formula is evident, especially in the low volatility scenario, since the errors are in the order of basis points or even lower, so low enough for applications in the market.
Additionally, Figure 2.2 shows the evolution of the option prices and their difference (defined as *Diff* = *Approximated Value* - *Exact Value*) as the strike varies. The panels on the left hand side show the low volatility scenario ($\nu = 0.15$), the ones on the right show the high volatility case ($\nu = 0.3$). Then, the upper panels show the short maturity case ($T = 0.25$), the lower ones the long maturity scenario ($T = 5$ years). The panels prove that, for every case and strike, the difference between the prices is very small. It increases in the long maturity scenario and for far in-the-money and out-of-the-money strikes, even though it is still considerably smaller than the bid-ask spreads in the market. This confirms, once again, the accuracy of the approximated formula.

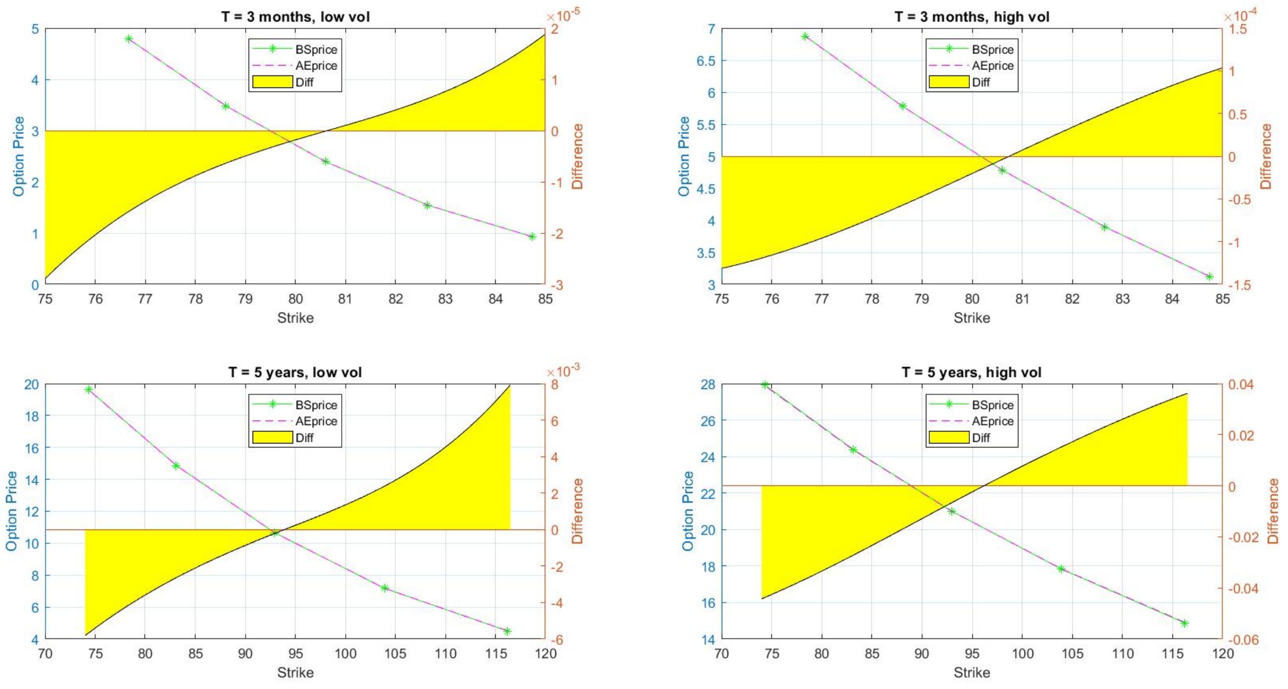


Figure 2.2: Comparison of the European Call Black&Scholes and Wiener-Itô chaos third order expansion prices and their errors

2.3. The approximation formula for single barrier options

Funahashi and Higuchi in [29] evaluated the asymptotic expansion formula of call and put barrier options of all four types, among which the Up-and-In one, with a sum truncation at the second order. They approximated the underlying asset dynamic with a polynomial of the Wiener process. Therefore, the probability density function required is the one of the Wiener process. After the application of Girsanov's theorem and the reflection principle, the asymptotic closed formula of the Up-and-In single barrier option is in Theorem 2.3 [26] below, whose proof is located in [29]:

Theorem 2.3. *The value of an Up-and-In barrier option with barrier level B, maturity T and strike K is approximated by:*

$$\begin{aligned} UI(T,K) = e^{-\int_0^T r(s)\,ds}\Bigg[&\frac{e^{\Omega_T}}{2\Sigma_T\sqrt{2\pi}}\left(e^{-\frac{(\omega_T^1(K)-\dot{\omega_T})^2}{2T}}X_1(T) - e^{-\frac{(\omega_T^1(B)-\dot{\omega_T})^2}{2T}}X_2(T)\right) \\ &+ \frac{e^{\Omega_T}}{2\Sigma_T}X_3(T)\left(\Phi\left(\frac{\omega_T^1(B)-\dot{\omega_T}}{\sqrt{T}}\right) - \Phi\left(\frac{\omega_T^1(K)-\dot{\omega_T}}{\sqrt{T}}\right)\right) \\ &+ \frac{F(0,T)e^{-\frac{\overline{B}^2}{2\Sigma_T}}}{\sqrt{2\pi}\Sigma_T^{\frac{5}{2}}}(\overline{B}^2(\overline{B}+\overline{K}_T)q(T) - \overline{K}_T q(T)\Sigma_T + \Sigma_T^3) \\ &+ F(0,T)\overline{K}_T\left(1-\Phi\left(\frac{\overline{B}}{\sqrt{\Sigma_T}}\right)\right)\Bigg] \end{aligned} \tag{2.35}$$

where $\overline{K}_T = 1 - \frac{K}{F(0,T)}$, $\overline{B} = \frac{B}{F(0,T)} - 1$, and $\Phi(x)$ is the cumulative distribution. $q(t)$,Σ_T,$\omega_t^1(B)$, Ω_T, $\dot{\omega_T}$ and $X_i(T)$ are defined in Appendix B.

Similarly to the Up-and-In barrier option, all the other options are derived. Given the maturity T and the strike K, for the Down-and-In barrier option the formula is the following [29]:

$$DI(T,K) = \frac{e^{\Omega_T}}{2T^2\Sigma_T}\left[f_1(T) + f_2(T)\left[1 + \frac{\ddot{\omega_T}}{\sqrt{\ddot{\omega_T}^2}}\left(2\Phi\left(\sqrt{\frac{\ddot{\omega_T}^2}{T}}\right) - 1\right)\right]\right], \tag{2.36}$$

where $\Phi(x)$ is the cumulative distribution and Σ_T, Ω_T, $\ddot{\omega_T}$, $f_1(T)$ and $f_2(T)$ are defined in Appendix B.

Eventually, the Up-and-Out and the Down-and-Out pricing formulae are obtained with the relationship of barrier options [29]:

$$C(T,K) = UI(T,K) + UO(T,K),$$
$$C(T,K) = DI(T,K) + DO(T,K),$$

where $C(T,K)$ indicates the European Call price computed with Eq.(2.33), $UI(T,K)$ is evaluated with Eq.(2.35) and $DI(T,K)$ is computed with Eq.(2.36). $UO(T,K)$ and $DO(T,K)$ stand for the Up-and-Out price and the Down-and-Out one, respectively.

3 | Chapter 3: Artificial Neural Networks

This chapter provides a brief introduction on artificial neural networks. This explanation is tailored to have a better understanding of the concepts that will be mentioned in the following chapters. The main sources of this chapter are [14, 15] and [26].

Artificial neural networks take their name from the human brain mesh that connects the neurons. Indeed, they are a mathematical model that imitates this complex structure. Originally, McCulloch and Pitts [50] proposed the first ANN model, called a formal neuron. Then Rosenblatt [52] evolved this concept to present the perceptron, which could only solve linear problems. This limitation was overcome by Rumelhart *et al.* [54], who defined the algorithm of back-propagation, which allowed the neural networks to learn in order to solve non-linear problems. Afterwards, research has progressed at a fast pace and ANN have been applied in many fields, including option pricing. [39, 41, 46] and their references therein offer some examples.

The architecture of a neural network is usually composed of an input layer, some hidden layers and the output layer. The number of layers and nodes per layer is chosen depending on the problem. In this thesis the number of layers varies between 3,5 or 7, while the number of neurons is 16 or 64 per each layer. Moreover, as in most cases, the output layer has only one node. This occurs when there is a single labelled quantity y_d to compare the estimated value $\hat{y_d}$ with for each d^{th} input-output pair $(\mathbf{x}_d, y_d)$ of the data set X.

In this thesis the neural network adopted are of the simplest type: the feed-forward neural networks. Indeed, their connections between the units do not form cycles and information travels only in one direction in each phase of the back-propagation algorithm. In the forward phase, calculations begin in the input and end in the output layer, *viceversa* occurs in the backward one.

This chapter is structured as follows: Section 3.1 examines the back-propagation algorithm, Section 3.2 provides a description of the neural network hyper-parameters and Section 3.3 describes briefly how such hyper-parameters are tuned to obtain optimal results.

3.1. Back-propagation algorithm

First of all, it is useful to define all the terms that are mentioned in the algorithm.

- $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_d, y_d), ..., (\mathbf{x}_N, y_N)\}$ is the set of input-output pairs to pass to the neural network. In this case it is assumed there is a scalar output for each pair;
- o_j^l is the output of the j^{th} node in the l^{th} layer;
- $\hat{y}_d$ is the estimated output, i.e. the output of the output layer;
- r_l is the number of nodes in the l^{th} layer;
- $w_{i,j}^l$ is the weight which measures the strength of the connection of the i^{th} node in the $(l-1)^{th}$ layer with the j^{th} node in the l^{th} layer;
- b_j^l is the bias of the j^{th} node in the l^{th} layer. For simplicity, it is assumed that it is incorporated in the weights as $w_{0,j}^l$, i.e. $b_j^l = w_{0,j}^l$, fixing the output of the previous layer as $o_0^{l-1} = 1$;
- θ collectively represents the weights and the biases of the neural network;
- a_j^l is the activation of the j^{th} node in the l^{th} layer. It is formally defined as: $a_j^l := \left(\sum_{k=1}^{r_{l-1}} w_{k,j}^l o_k^{l-1}\right) + b_j^l = \left(\sum_{k=0}^{r_{l-1}} w_{k,j}^l o_k^{l-1}\right)$, due to $b_j^l = w_{0,j}^l$;
- f is the activation function. For simplicity it is assumed that it is the same for both hidden and output layers;
- $E(X, \theta)$ is the total error function, while E_d (or, equally, E) indicates the error function for the input-output pair $(\mathbf{x}_d, y_d)$.

The objective of the back-propagation algorithm is to determine the set of weights and biases of the neural network for which the total error function is minimised. For this reason, the algorithm makes use of the gradient descent (or a more efficient alternative of it) in order to evaluate the best set θ. More precisely, each iteration of the back-propagation algorithm which uses gradient descent has four steps:

- **Forward phase**;
- **Backward phase**;
- **Evaluation of the total gradient**;
- **Update of the weights**.

Each of these steps is treated in the following subsections.

3.1.1. Forward phase

At the end of this first back-propagation step these quantities are evaluated and stored: the estimated output $\hat{y}_d$, the activation a_j^l and the output o_j^l. These values are retrieved for each pair in X and for each node from the start. The input layer normally outputs the input itself, i.e. $o_i^1 = x_i^0$, where x_i^0 is the input received by the i^{th} node of the input layer and o_i^1 is the output of the input layer for the nodes of the first hidden layer. For these and the nodes in the following layers the calculation of the output is different. Indeed, every i^{th} node in the $(l-1)^{th}$ layer is coupled with the j^{th} node of the l^{th} layer. The output o_j^l of the j^{th} node is expressed by $o_j^l = f(a_j^l)$, where a_j^l is the activation evaluated as per its definition. Then, the output o_j^l will be the input for the nodes in the subsequent $(l+1)^{th}$ layer. Eventually, the output layer node will produce the estimated output $\hat{y}_d$.

3.1.2. Backward phase

In this phase the main objective is to evaluate and store the term $\frac{\partial E_d}{\partial w_{i,j}^l}$. This procedure takes its name from the fact that these computations are executed from the output to the input layer. From now on the subscript d will be omitted in this section for simplicity. First of all, the chain rule is applied to the error function derivative, resulting in:

$$\frac{\partial E}{\partial w_{i,j}^l} = \frac{\partial E}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{i,j}^l},$$

where a_j^l is the activation. This expression can be rewritten by evaluating each term. The first one is called error:

$$\delta_j^l := \frac{\partial E}{\partial a_j^l}.$$

The second term is calculated using the definition of activation:

$$\frac{\partial a_j^l}{\partial w_{i,j}^l} = \frac{\partial}{\partial w_{i,j}^l}\left(\sum_{k=0}^{r_{l-1}} w_{k,j}^l o_k^{l-1}\right) = o_i^{l-1}.$$

In the end, the alternative expression of the partial derivative of the error function over the weight is:

$$\frac{\partial E}{\partial w_{i,j}^l} = \delta_j^l o_i^{l-1}.$$

In this thesis the error function chosen is the halved mean squared error:

$$E(X,\theta) = \frac{1}{2N}\sum_{d=1}^{N}(\hat{y}_d - y_d)^2 = \frac{1}{N}\sum_{d=1}^{N} E_d,$$

where y_d is the desired output and $\hat{y_d}$ is the estimated one, given the pair $(\mathbf{x}_d, y_d)$ of the data set X. In this particular case the derivative can be rewritten in a different expression. In the output layer, the halved mean squared error is:

$$E = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(f(a_1^L) - y)^2,$$

given f as the activation function. Then the error term is:

$$\delta_1^L = (f(a_1^L) - y)f^{'}(a_1^L) = (\hat{y} - y)f^{'}(a_1^L).$$

So the derivative is:

$$\frac{\partial E}{\partial w_{i,1}^L} = \delta_1^L o_i^{L-1} = (\hat{y} - y)f^{'}(a_1^L)o_i^{L-1}.$$

For any other l^{th} layer with $1 \leq l < L$, the error term of the j^{th} node is:

$$\delta_j^l = \sum_{k=1}^{r_{l+1}} \frac{\partial E}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial a_k^l} = \sum_{k=1}^{r_{l+1}} \delta_k^{l+1} \frac{\partial a_k^{l+1}}{\partial a_k^l}.$$

Reminding that

$$a_k^{l+1} = \sum_{j=1}^{r_l} w_{j,k}^{l+1} f(a_j^l),$$

then the partial derivative of a_k^{l+1} over a_k^l is

$$\frac{\partial a_k^{l+1}}{\partial a_k^l} = w_{j,k}^{l+1} f^{'}(a_j^l).$$

Therefore, the error term δ_j^l is:

$$\delta_j^l = \sum_{k=1}^{r_{l+1}} \delta_k^{l+1} w_{j,k}^{l+1} f^{'}(a_j^l).$$

Given this, each derivative of the error function E over the weight results in

$$\frac{\partial E}{\partial w_{i,j}^l} = \delta_j^l o_i^{l-1} = f^{'}(a_j^l) o_i^{l-1} \sum_{k=1}^{r^{l+1}} w_{j,k}^{l+1} \delta_k^{l+1}$$

and it is stored for the next step. This equation above gives the name to the algorithm: back-propagation is the short term for "backward propagation by errors" and the last equation shows that the error δ_j^l is always dependent on the error δ_k^{l+1} in the following layer.

3.1.3. Evaluation of the total gradient

Once all layers have been considered for each $(\mathbf{x}_d, y_d)$ pair, then the total error derivative is computed. This value, independently on the choice of the error function, is the sum of the derivatives of the error terms of each individual input-output pair with respect to the weights. Indeed:

$$\frac{\partial E(X,\theta)}{\partial w_{i,j}^l} = \frac{1}{N}\sum_{d=1}^{N} \frac{\partial}{\partial w_{i,j}^l}\left(\frac{1}{2}(\hat{y}_d - y_d)^2\right) = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial E_d}{\partial w_{i,j}^l}.$$

3.1.4. Update of the weights

Eventually the weights are updated using the gradient descent formula:

$$\Delta w_{i,j}^l = -\alpha \frac{\partial E(X,\theta)}{\partial w_{i,j}^l},$$

where α is the learning rate, a parameter which indicates the size of the step in the negative direction of the gradient. Due to the assumption that the biases are incorporated in the weights, the update is done only on them. Otherwise, the biases are calibrated too.

This concludes one iteration of the back-propagation algorithm. These four steps are repeated until a local minimum of the error function is reached or a convergence criterium is met. Once this occurs, then the neural network has learnt the best mapping. This is the one that best approximates the function that returns the output y_d, given the paired input $\mathbf{x}_d$. Thus, the result of this mapping is the best approximation of the desired output, i.e. $\hat{y}_d$.

3.2. Designing an Artificial Neural Network

As mentioned in the introduction of this chapter, the neural network is customisable. For example, the architecture of the network can be changed by selecting a different number of layers or nodes per layer. Beside this, there are other elements that can be varied to avoid underfitting or overfitting. These components are defined and discussed in this section.
First of all, the activation function is quite crucial in the creation of the ANN. Usually, its choice (at least for the output layer) is determined by the type of problem to be modelled [8]: a classification one needs an activation function that has a dichotomous outcome,

otherwise in a regression one it should return a real value from a continuous range. In this thesis the problem belongs to the latter category and in order to replicate the results of [26], the two activation functions chosen are the following:

- ReLU: $f : \mathbb{R} \mapsto \mathbb{R}$ with $f(x) := \max(0, x)$ and $x \in \mathbb{R}$ [8];
- Softmax: $f : \mathbb{R}^K \mapsto (0,1)^K$ with $f(x)_i := \frac{e^{x_i}}{\sum_{i=0}^{K} e^{x_i}}$ and $\mathbf{x} \in \mathbb{R}^K$ [8].

Another important aspect is the choice of the optimizer. In Section 3.1 it is described the back-propagation which uses gradient descent. This is the simplest optimization algorithm to understand. However, in practice, it is not the most computationally efficient and research has examined different alternatives of it. One of them is the Adam algorithm, first proposed in [43]. In brief, the weights and biases are calibrated at each iteration depending on the exponential moving averages of the gradient and the squared gradient. These quantities are the bias-corrected estimates of the first moment (the mean) and the second moment (the uncentred variance) of the gradient. This is the optimizer that has been chosen for this thesis. It has many great properties: it requires little memory since it evaluates only the first order gradients, it is an adaptive algorithm that performs well in many problems and it does not need much hyper-parameter tuning compared to other optimization algorithms [43].
Indeed, the choice of the optimizer has a great influence on the hyper-parameters required. For example, the Adam optimizer requires four tuning parameters: the learning rate α, the exponential decay rates β_1 and β_2 for, respectively, the first and the second moment estimates and the learning rate decay λ [17]. Among them, the learning rate is the most relevant, since it is an upperbound for the magnitude of the step size of each iteration [43] and has a great impact on the computation time [48].
Even though they are not parameters of the Adam optimizer, batch size and the number of epochs allow to customise the neural network too. In the default case of the gradient descent or Adam, the update of the weights occurs at the end of each epoch, i.e. once all the input-output pairs of the training data are run. The batch size allows to adjust how many optimisations are done in one epoch. Indeed, given a training data set of N paired samples, then $\frac{N}{batch\ size}$ is the number of weight updates for each epoch. Therefore, the total number of calibrations is given by $\frac{\#\ epochs \times N}{batch\ size}$ [6]. In practice, usually the number of epochs stays fixed *a priori* and the batch size can be modified to improve the performance [59]. Its values are powers of 2: lower the value, the higher the regularisation effect and lower the memory capacity required, but the downside is that the gradient convergence to a minimum is less direct [56].

3.3. Optimizing an Artificial Neural Network

The choice of the hyper-parameters impacts the performance of the ANN. For this reason, often machine learning practitioners employ tuning techniques to obtain optimal results from their model. Their main advantages are [2]:

- they reduce the human effort necessary for applying machine learning, since the tuning procedure allows to evaluate the performance of several different models automatically;
- they improve the performance of machine learning algorithms (by tailoring them to the problem at hand) [51, 60];
- they improve the reproducibility and fairness of scientific studies [11, 55].

The simplest criterium to determine the best configuration of hyper-parameters is based on the minimisation of the loss function. The best set is the one for which the loss function is minimised the most at the end of the optimization algorithm. However, there are many tuning techniques, even more complex, which can be implemented.
Moreover, there is an immense variety of algorithms that indicate how to search the optimal configuration in the given range of the hyper-parameters. In grid search, for example, every combination of every hyper-parameter in the specified subset is evaluated. This algorithm is easily parallelisable, but it suffers by the curse of dimensionality [45]. Another example is random search. In this case, only some configurations, selected randomly from the search space, are tested [45]. This method, according to [10], is more computationally efficient than grid search.
The scheduler and the stopper are other means to a less computationally expensive tuning phase. The scheduler can early terminate, pause or clone trials, i.e. the tests of hyper-parameters configurations. The one used in this thesis is the asynchronous successive halving (ASHA) scheduler [44]. It is based on the assumption that if the configuration trial performs well over an initial short time interval, then it will perform well at longer time intervals. In brief, it early stops the predefined portion of the least performing trials, after a specified number of training iterations. The stopper stops the trial as soon as the loss function reaches a plateau, i.e. does not decrease further. This allows to avoid over-fitting.

4 | Chapter 4: Numerical Procedure

In this chapter, the procedure to compare the results of two pricing methods is described in detail. Both of them make use of the same Neural Network model, except for the information that they receive as input. Indeed, the first one uses as label the analytic (or semi-analytic) option price, while the second one employs the difference between the analytic (or semi-analytic) and the asymptotic expansion approximated price. These are referred as method 1 and 2, respectively, throughout this chapter. These two methods are compared across multiple aspects, by varying the model of the underlying dynamics (i.e. Black & Scholes, Heston *etc.*), the option to price (i.e. European Call, Up-and-In barrier or the Down-and-In barrier) and the architecture and other hyper-parameters of the Neural Network.
The methodology followed is very similar to the one of Funahashi in [26]. It can be summarized into four steps: creation of the data sets, building of the neural networks, tuning of the hyper-parameters and comparison of the results of the two methods.

4.1. Creation of the data sets

This phase includes four consecutive steps: the simulation of the parameters, the evaluation of the option price with the analytic or semi-analytic formula, and with the asymptotic expansion formula described in Chapter 2 and the standardization of the input features and the label for the neural networks.

4.1.1. Simulation of the parameters

Each financial model requires a different set of parameters. In the following paragraphs, their ranges are specified in each scenario dealt in this thesis. The exact subdivision of cases is listed in Table 4.1 and it is referred in the whole chapter coherently.

Case	Input vector	Label (method 1)
0	$[S_0, r, \epsilon, K]$	B&S EU Call price
1	$[S_0, r, \epsilon, K]$	B&S EU Call price
2	$[S_0, r, \epsilon, K]$	B&S EU Call price
3	$[S_0, r, \epsilon, K]$	B&S EU Call price
4	$[S_0, r, \epsilon, K]$	B&S EU Call price
5	$[S_0, r, \epsilon, K]$	B&S EU Call price
6	$[S_0, r, \epsilon, K]$	B&S EU Call price
7	$[S_0, r, \epsilon, K]$	B&S EU Call price
8	$[S_0, r, \epsilon, K]$	B&S EU Call price
9	$[S_0, r, \epsilon, K]$	B&S EU Call price
10	$[S_0, r, \epsilon, K]$	B&S & Eq.(4.1) EU Call price
11	$[S_0, r, \epsilon, K]$	B&S EU Call Delta
12	$[S_0, r, \epsilon, K]$	B&S EU Call Vega
13	$[S_0, \epsilon]$	B&S EU Call Vega ATM
14	$[r, T, S_0, \nu_0, \rho, \epsilon, \kappa, \theta]$	Heston Up-and-In price
15	$[r, T, S_0, \nu_0, \rho, \epsilon, \kappa, \theta]$	Heston Up-and-In price
16	$[T, S0, \beta, \epsilon]$	CEV Down-and-In price
17	$[T, S0, \alpha, \beta, \mu]$	NLV Down-and-In price

Table 4.1: Summary of the details to reproduce each case. The label of method 2 is always the difference between the label quantity of method 1 and the respective asymptotic expansion approximated value.

- **Black & Scholes** The generation of the Black & Scholes parameters differs depending on the case taken into account. In the first scenario (cases 1-8), the Black & Scholes European Call price and its difference with the one obtained with Eq.(2.2) are the labels of neural networks of, respectively, method 1 and method 2. The input vector is the same for both methods and it consists of $[S_0, r, \epsilon, K]$. Table 4.2 summarises how the parameters are generated.
  In the second scenario (cases 11 and 12), the Greeks and their difference with the ones obtained with Eq.(2.2) are the labels of neural networks of, respectively, method 1 and method 2. In particular, the Delta and the Vega of a European Call option are analysed. The input vector is the same as the one in the first scenario. The parameters are described in Table 4.3.

In the third scenario (case 13), the label for the neural network of method 1 is the Vega around the *at-the-money* scenario. Here, the unstable behaviour of this Greek is evaluated. Naturally, the difference of this quantity with the one computed with the differentiation of Eq.(2.2) with respect to ϵ is the label of the neural network of method 2. In this case, the input vector consists of $[S_0, \epsilon]$, since r and K are fixed. Table 4.4 below reports the parameters used.

The fourth scenario (case 9) is analogous to the first one for what concerns the labels and input vector. However, its parameters are mostly fixed to show the evolution of the difference of the prices with respect to the varying strike. Table 4.5 lists this choice of parameters.

The last scenario (case 10) compares the predicted European Call prices of both methods with the European Call price computed with the approximated formula of Eq.(4.1). In this case the parameters are the same as in the first scenario.

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	[1,5]	Random integer
Initial asset price	S_0	[50,150]	Random, uniformly distributed
Short interest rate	r	[0,0.05]	Random, uniformly distributed
Volatility	ϵ	[0.01,1]	Random, uniformly distributed
Uniform variable	U	[0.8,1.2]	Random, uniformly distributed
Strike	K	[40,180]	$S_0 \times U$

Table 4.2: Parameters used to evaluate the Black & Scholes price and the one obtained with the asymptotic expansion formula (Eq.(2.2)).

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	[1,5]	Random integer
Initial asset price	S_0	[80,150]	Random, uniformly distributed
Short interest rate	r	[0.00001,0.1]	Random, uniformly distributed
Volatility	ϵ	[0.01,1]	Random, uniformly distributed
Strike	K	[64,180]	$S_0 \times U$

Table 4.3: Parameters used to evaluate the Black & Scholes Delta and Vega and the one obtained with the differentiation of the asymptotic expansion formula (Eq.(2.2)).

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	1	Fixed
Initial asset price	S_0	[95,105]	Random, uniformly distributed
Short interest rate	r	0	Fixed
Volatility	ϵ	[0.001,0.1]	Random, uniformly distributed
Strike	K	100	Fixed

Table 4.4: Parameters used to evaluate the Black & Scholes Vega and the one obtained with the differentiation of the asymptotic expansion formula (Eq.(2.2)) around the *at-the-money* scenario.

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	5	Fixed
Initial asset price	S_0	80	Fixed
Short interest rate	r	0.03	Fixed
Volatility	ϵ	0.3	Fixed
Strike	K	[50,150]	Random, uniformly distributed

Table 4.5: Parameters used to evaluate the Black & Scholes price and the one obtained with the asymptotic expansion formula (Eq.(2.2)), specifically in the fourth scenario (case 9).

- **Heston** For cases 14 and 15, Table 4.6 summarises the parameters used to price a barrier option under the Heston model, which is defined in Section 1.2. The input random vectors of the neural networks consist of $[r, T, S_0, \nu_0, \rho, \epsilon, \kappa, \theta]$.

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	[0,2]	Random, uniformly distributed
Initial asset price	S_0	[80,120]	Random, uniformly distributed
Short interest rate	r	[0,0.05]	Random, uniformly distributed
Initial variance	ν_0	[0,1]	Random, uniformly distributed
Correlation between the Wiener processes	ρ	[-0.999,0.999]	Random, uniformly distributed
Volatility of volatility	ϵ	[0.001,0.15]	Random, uniformly distributed
Mean reversion rate	κ	[0.01,4]	Random, uniformly distributed
Uniform variable 1	U_1	[0.8,1.2]	Random, uniformly distributed
Uniform variable 2	U_2	[1.001,1.1]	Random, uniformly distributed
Uniform variable 3	U_3	[0.5,2]	Random, uniformly distributed
Strike	K	[64,144]	$S_0 \times U_1$
Barrier level	B	[80.08,132]	$S_0 \times U_2$
Long-run average variance times the mean reversion rate	θ	[0,2]	$\nu_0 \times U_3$

Table 4.6: Parameters used to evaluate the price of a Up-and-In barrier option under the Heston model.

- **CEV** For case 17, Table 4.7 indicates the parameters used to price a Down-and-In single barrier option under the CEV model, defined by the Eqs. (1.8) and (1.9). The input vector is composed of $[T, S0, \beta, \epsilon]$.

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	[0.25,5]	Random, uniformly distributed
Initial asset price	S_0	[80,120]	Random, uniformly distributed
Short interest rate	r	0	Fixed
Elasticity factor	β	[0.65,0.85]	Random, uniformly distributed
Uniform variable 1	U_1	[0.15,0.25]	Random, uniformly distributed
Uniform variable 2	U_2	[0.8,1.1]	Random, uniformly distributed
Uniform variable 3	U_3	[0.95,0.999]	Random, uniformly distributed
Volatility	ϵ	[0.29,1.34]	$S_0^{(1-\beta)} \times U_1$
Strike	K	[64,132]	$S_0 \times U_2$
Barrier level	B	[76,119.88]	$S_0 \times U_3$

Table 4.7: Parameters used to evaluate the price of a Down-and-In barrier option under the CEV model.

- **Non linear volatility model** For case 18, Table 4.8 indicates the parameters used to price a Down-and-In single barrier option under the non linear volatility model, defined by the Eqs.(1.10) and (1.11). The input vector is composed of $[T, S0, \alpha, \beta, \mu]$.

Variable	**Symbol**	**Range**	**Generation**
Maturity	T	[0.25,5]	Random, uniformly distributed
Initial asset price	S_0	[80,120]	Random, uniformly distributed
Short interest rate	r	0	Fixed
Parameter 1	α	[0.01,0.1]	Random, uniformly distributed
Parameter 2	β	[0.005,0.085]	Random, uniformly distributed
Parameter 3	μ	[-0.05,-0.15]	Random, uniformly distributed
Uniform variable 1	U_1	[0.8,1.1]	Random, uniformly distributed
Uniform variable 2	U_2	[0.95,0.999]	Random, uniformly distributed
Strike	K	[64,132]	$S_0 \times U_1$
Barrier level	B	[76,119.88]	$S_0 \times U_2$

Table 4.8: Parameters used to evaluate the price of a Down-and-In barrier option under the non linear volatility model.

4.1.2. Evaluation of the option price with the analytic or semi-analytic formula

Once the parameters are set, then the quantities which will serve as labels in the neural networks should be calculated. In this subsection, the computation of the benchmark values is analysed in depth, while the details for the calculations of the asymptotic expansion approximations are referred in Subsection 4.1.3. The computation of these quantities varies according to the model and the scenario chosen.

- **Black & Scholes** For this model there are analytic formulae to evaluate the required quantities. Consider the same scenarios as listed in Section 4.1.1. In the first and fourth scenarios (cases 1-9), the European Call price is evaluated according to Eq.(1.12). In the second and third ones (cases 11-13), the Greeks are computed with the Eqs. (1.20) and (1.21). In the last one (case 10), the European Call price is calculated with the following formula:

$$C = e^{-rT} \max\left(S_0 e^{rT} - K, 0\right). \tag{4.1}$$

- **Heston** The price of the Up-and-In barrier option under the Heston model (cases 14 and 15) is evaluated with Monte Carlo simulations and discretised with the QE scheme [5]. As in the previous case, the time interval has been subdivided 10^4 times, 10^5 simulated paths of the Wiener process have been realised and the Numba JIT compiler has been used. Furthermore, it is remarked that the Heston models defined in [5] and Section 1.2 differ. Indeed, the Heston model defined in the former source and used to evaluate the Monte Carlo prices is:

$$\frac{dS_t}{S_t} = r(t)dt + \sqrt{\nu_t}dW_t^S \tag{4.2}$$
$$d\nu_t = \kappa(t)(\overline{\theta}(t) - \nu_t)dt + \epsilon\sqrt{\nu_t}dW_t^\nu, \tag{4.3}$$

  which means that $\overline{\theta}(t) = \frac{\theta(t)}{\kappa(t)}$ must hold for the two model forms to be equivalent. Therefore, the long-run average variance $\overline{\theta}(t)$ is simulated according to this relationship with the given parameters $\theta(t)$ and $\kappa(t)$ set in Table 4.6.

- **CEV** The CEV model is adopted to price a Down-and-In barrier option. The method to obtain the option price is the Brownian Bridge Monte Carlo [33]. It is used for path-dependent options, such as barrier ones. Indeed, it evaluates the option price taking into account the probability that the value of the underlying touches the barrier level in a time instant between the initial time and the maturity.

Briefly, the probability of touching the barrier level B in a time instant $\tau_B \in (t_i, t_{i+1})$, given the asset values S_i and S_{i+1}, is [21]:
For $S_i > B$:

$$\mathbb{P}(\tau_B \in (t_i, t_{i+1})|S_i, S_{i+1}) = e^{\left[-\frac{2(ln(S_i)-ln(B))(ln(S_{i+1})-ln(B))}{\sigma^2_{CEV}dt}\right]}, \tag{4.4}$$

while for $S_i < B$:

$$\mathbb{P}(\tau_B \in (t_i, t_{i+1})|S_i, S_{i+1}) = e^{\left[-\frac{2(ln(B)-ln(S_i))(ln(B)-ln(S_{i+1}))}{\sigma^2_{CEV}dt}\right]}. \tag{4.5}$$

In the specific case of a Down-and-In option, it is assumed $S_0 > B$. Given this and Eq. (4.4), the probability that the asset value trajectory touches the barrier level at some instant $\tau_B \in [0, T]$ is [21]:

$$\mathbb{P}(\tau_B \in [0, T]|S_0, S_T = S_m) = 1 - \prod_{i=0}^{m-1}\left(1 - e^{\left[-\frac{2(ln(S_i)-ln(B))(ln(S_{i+1})-ln(B))}{\sigma^2_{CEV}dt}\right]}\right)^+.$$

In the end the price of the Down-and-In Call is obtained with [21]:

$$DI(T) = e^{-rT}(S_T - K)^+(\mathbb{P}(\tau_B \in [0, T]|S_0, S_T = S_m)).$$

For the complete derivation of these probabilities see [9].
Additional details of the Monte Carlo pricing under the CEV model are that the discretization of the underlying makes use of the Euler scheme, the time interval has 10^4 subdivisions and 10^5 simulated paths of the Wiener process have been computed. As for the financial models above, the Numba JIT compiler is adopted.

- **Non linear volatility model** The procedure to price the Down-and-In option under the non linear volatility model is identical to the one of the CEV model. The only difference is the volatility in Eq. (4.4) is the one defined in Eq.(1.10).

4.1.3. Evaluation of the option price with the asymptotic expansion formula

The computation of the prices varies according to the option taken into account. For the European Call option, the equations to obtain the approximated price are stated in Lemma 2.22, Theorem 2.2 and Appendix A. For the Up-and-In and the Down-and-In barrier options, the equations are Eqs.(2.35) and (2.36) and additional quantities are

defined in Appendix B.
Two remarks on the Up-and-In barrier option case are required. Firstly, the Heston model form to obtain the approximated prices is the one in Section 1.2, therefore the one proposed by Funahashi in [26] and previously in [29]. Secondly, sometimes the computation of the approximated price results in a NaN value, in a negative value or a completely different price to the Monte Carlo one. It is observed that this occurs since the integral to get $\Omega(t)$ tends to diverge sometimes. To overcome this issue, two techniques have been adopted:

- **Approximating the parameters**. At the end of each iteration it is checked whether the price is a NaN or a negative value or its difference with the Monte Carlo price is greater than 1 in absolute value. If this is the case, every parameter in Table 4.6 is truncated with one less final decimal and the computation is done all over again. If the difference between the prices does not decrease in absolute value, then the price got without approximating the parameters is stored, otherwise the new price is saved and the truncation is iterated until its difference worsens. If the new price is still a NaN or a negative value, then the truncation is iterated until a positive number is obtained or the decimals end.

- **Simulating few more prices than required**. This foresight has allowed to neglect the few remaining data instances with NaN prices and still have the input neural network data set with the pre-estabilished size.

Concerning the Down-and-In under the non linear volatility model, some Monte Carlo simulations resulted in a NaN value. For this reason, few more prices have been simulated.

4.1.4. Setting the neural network input features and labels

Once both option prices have been obtained, their difference is evaluated:

$$D(\xi) = C^S(\xi) - C^{\overline{S}}(\xi), \tag{4.6}$$

where $C^S(\xi)$ stands for the option price obtained with the analytic/semi-analytic formula, $C^{\overline{S}}(\xi)$ indicates the option price computed with the asymptotic expansion formula and ξ is the parameter vector required to compute the prices.
In brief, the first method is represented by the map $\mathscr{M}_C : \xi \mapsto C^S(\xi)$: the neural network takes the parameter vector as input and the benchmark price as label and returns the prediction of the benchmark price, indicated by $C^S_{ANN}(\xi)$. Whereas, the second method is indicated by $\mathscr{M}_D : \xi \mapsto D(\xi)$: the neural network takes the parameter vector as input and the difference between the prices as label and returns the prediction of the difference

of the prices $D_{ANN}(\xi)$. Then, the prediction of option price is attained by:

$$C^D_{ANN}(\xi) = C^{\overline{S}}(\xi) + D_{ANN}(\xi). \tag{4.7}$$

Eventually, the comparison of the two methods is done by comparing $C^S_{ANN}(\xi)$ and $C^D_{ANN}(\xi)$.
A relevant aspect of the asymptotic option pricing formula is that its difference obtained with Eq.(4.6) results in a smooth and differential function. This property allows both to improve the efficiency of the ANN and to stabilize the learning of the Greeks, such as Delta and Vega. Their stable computation is guaranteed by Hornik *et al.* [38]:

Theorem 4.1. *Let $\mathbb{N}^{\sigma}_{d0,d1}$ be the set of ANN with activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$, input dimension $d_0 \in \mathbb{N}$ and the output dimension $d_1 \in \mathbb{N}$. Let $F \in C^n$, and $F_{ANN} : \mathbb{R}^{d0} \mapsto \mathbb{R}$. Then, if the (non-constant) activation function is $\sigma \in C^n(\mathbb{R})$, then $\mathbb{N}^{\sigma}_{d0,d1}$ arbitrarily approximate F and all its derivatives up to order n.*

See, also [39, 41].
Furthermore, higher differentiations of Eq.(2.33) are accurate as well. For example, an ulterior differentiation of delta results in a precise approximation of Gamma [31].
The input vectors and the labels differ according to the model used and the scenario taken into account: see Table 4.1 to get to know them in depth for each case. However, they have a common aspect: they are standardised before being passed to the neural network. Both features and labels are scaled with the standard scaler, which removes the mean and scales to unit variance. In the end, the predictions are rescaled back to compare the predicted results with the true labels.

4.2. Building the artificial neural networks

The same cases listed in Table 4.1 are referred in Table 4.9 to state the aspects of the neural networks considered. These are the data set dimension, the number of layers, the number of nodes per layer and the activation function.
The data set was split into training, validation and test sets according to the following proportions: 70, 15 and 15%. This proportion is a popular choice especially when the data set size is only 10^3. Actually, Funahashi in [26] does not mention the validation set and adopts the popular 80/20% split for the training and test set. However, in this thesis it was preferred to include the validation set to have more insight on the hyperparameters.

Case	Set size	# layers	# nodes per layer	Act. function
0	10^3	3	16	ReLU
1	10^3	5	64	ReLU
2	10^3	7	64	ReLU
3	10^4	3	16	ReLU
4	10^4	5	64	ReLU
5	10^4	7	64	ReLU
6	10^3	3	16	softmax
7	10^4	3	16	softmax
8	10^4	5	16	softmax
9	10^3	3	16	ReLU
10	10^3	3	16	ReLU
11	10^3	5	64	ReLU
12	10^3	5	64	ReLU
13	10^3	3	64	softmax
14	10^3	3	64	ReLU
15	10^3	7	64	ReLU
16	10^3	3	16	ReLU
17	10^3	3	16	ReLU

Table 4.9: Summary of the details to build the neural networks.

4.3. Tuning the artificial neural network

All cases are examined twice: first they are executed with fixed hyper-parameters and then with optimized ones. For the first case, the same hyper-parameters hold for the models of both method. For the latter scenario, they vary for each method and each case. The tuning phase has been implemented by using the Tune library of Ray [1]. Around 200 configurations have been sampled randomly from the search space defined in Table 4.10. For reproducible results, every configuration is associated with a seed. Moreover, the maximum number of epochs is set to 20, even though most trials terminate early due to the stopper or the ASHA scheduler. Concerning this object, it activates itself after 2 training iterations (grace period) and the proportion of trials that are periodically early stopped is $\frac{1}{2}$ of the total remaining trials (reduction factor).

Hyper-parameter	**Range**
Nodes per each layer	{4,8,16,32,64,128}
Learning rate	Log-uniform in $(10^{-4}, 10^{-1})$
Batch size	{2,4,8,16}
Seed	Integer in $(0,10^6)$

Table 4.10: The search space of hyper-parameters. The number of nodes is selected separately for each layer. The seed is included to ensure reproducibility of configurations.

4.4. Comparison of the methods

The predicted quantities used to compare the two methods are respectively $C^S_{ANN}(\xi)$ and $C^D_{ANN}(\xi)$. Each method is evaluated by analysing the test set data according to its prediction accuracy, stability, robustness and speed of convergence. These aspects are measured with the following quantities:

- **Root Mean Squared (Prediction) Error (RMSE)**. It is a measure of error defined as [37]:

  $$RMSE := \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

  where n is the sample dimension, y_i is the true label and $\hat{y}_i$ is the corresponding prediction. Lower the RMSE, higher the predictive accuracy of the model. It is preferred especially when errors are normally distributed and when error outliers should be penalised, as it tends to overweight large errors.

- **Mean Absolute Error (MAE)**. It is a measure of error defined as [37]:

  $$RMSE := \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

  where n is the sample dimension, y_i is the true label and $\hat{y}_i$ is the corresponding prediction. Lower the MAE, higher the predictive accuracy. It is preferred when errors have a Laplacian distribution and when a robust indicator is desired, as it tends to weight equally small and large errors.

- **Mean and Variance of the error density**. They give insight on the error distribution. In particular, the mean value indicates whether there is a tendency to overestimate, if positive, or underestimate, if negative, the true value. The variance indicates the dispersion of the error. Optimally, the mean and the variance should be close to zero. These quantities assess the prediction stability over the range of the true targets [22].

For every case, the predicted prices are displayed in a scatter plot with their true benchmark price on the x axis. Moreover, the density of the difference between the predicted value and the true benchmark price is showed in a histogram. These plots are associated with the mean and the variance of the error distributions. Furthermore, the computation times of the neural networks are other means of comparison.

5 | Chapter 5: Results Discussion

In this chapter the results of the two methods described in Chapter 4 are discussed. The seed set to ensure reproducibility of the outcomes is 190598. The characteristics of the PC adopted for the offline computations are: Processor Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz; RAM 8.00 GB; Operating System Windows 10 Pro.
Following the subdivision of the cases according to Table 4.1 and 4.9, this chapter is organised as follows: in Section 5.1 the cases 0-10 are analysed; next, in Section 5.2, the cases 11-13 on the Greeks are discussed; subsequently, the Up-and-In barrier option cases 14 and 15 are examined in Section 5.3; eventually, case 16 and 17 are treated in Section 5.4 and 5.5, respectively.

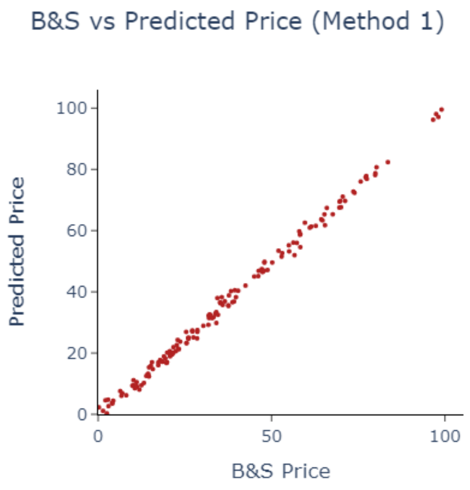
5.1. Results of the EU Call under Black & Scholes

In this section, all cases have in common that the label of the first method is the European Call price under the Black & Scholes model. Accordingly, the label of the second method is the difference between the B&S price and its asymptotic approximation.
The computation times of simulating the prices 7.8850×10^{-4} seconds for the B&S price only, 0.1159 seconds to simulate the asymptotic approximations too.

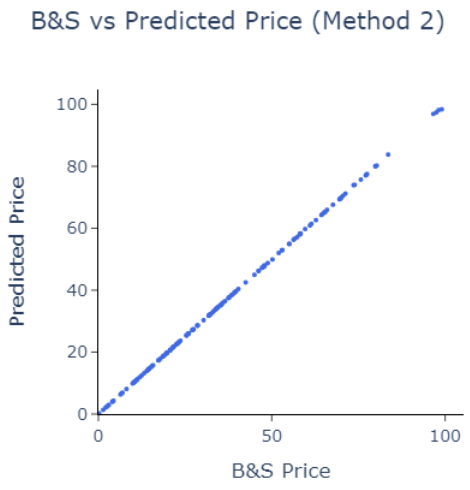
5.1.1. Using the same hyper-parameters: Black & Scholes

For what concerns the manually fixed hyper-parameters, for every case the optimizer is Adam, the batch size is 2 and the number of epochs is 15. For cases 0 and between 7 and 10, the learning rate is 0.001, for cases between 1 and 5 it is 0.0001 and for case 6 it is 0.002. The other Adam algorithm parameters are set to their default values.
The results show an evident superiority of the second method compared to the first.
Method 2 offers more accurate predictions than method 1. This is shown in Figure 5.1 and 5.2, where the predictions C^S_{ANN} and C^D_{ANN} on the y-axis are compared with the true B&S prices C^S on the x-axis. The scatter plot indicates a more accurate bisector for C^D_{ANN} vs. C^S. To quantify the predictive precision of the methods, Table 5.3 lists the RMSEs and the MAEs. For every case, these measures of errors are lower for method 2

and this confirms the accuracy shown in Figure 5.1 and 5.2.

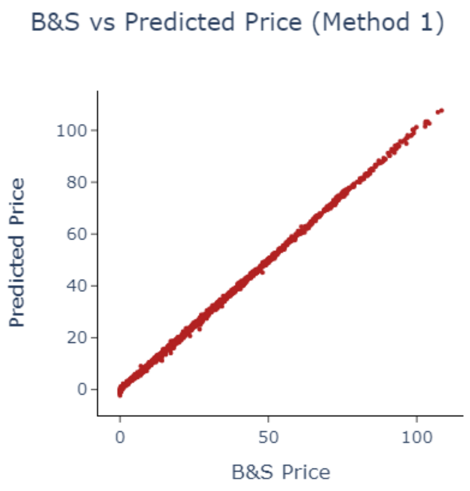


(a) Method 1.

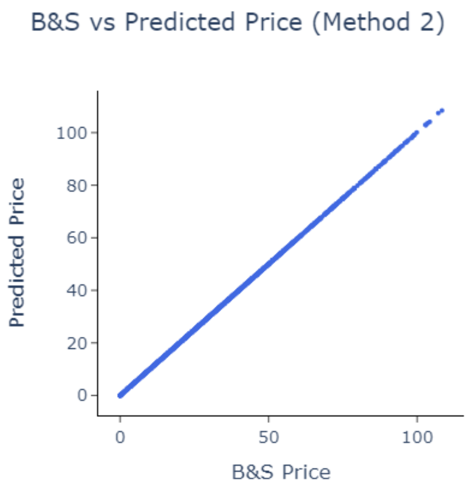


(b) Method 2.

Figure 5.1: Comparison of the Black & Scholes price on the x-axis and the predicted prices C^S_{ANN} (left) and C^D_{ANN} (right) on the y-axis with 10^3 samples, 3 layers and 16 nodes per layers (case 0).

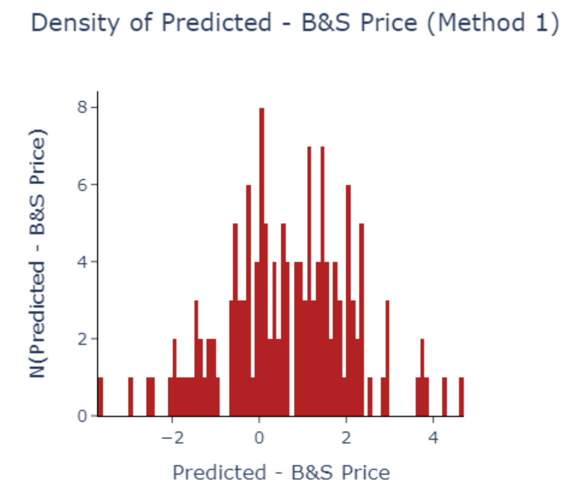


(a) Method 1.

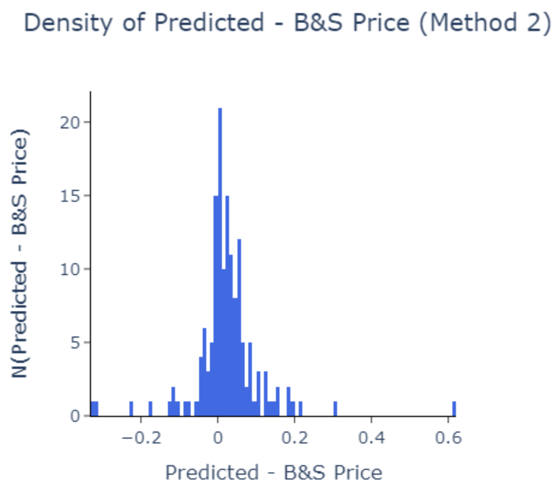


(b) Method 2.

Figure 5.2: Comparison of the Black & Scholes price on the x-axis and the predicted prices C^S_{ANN} (left) and C^D_{ANN} (right) on the y-axis with 10^4 samples, 5 layers and 64 nodes per layers (case 4).

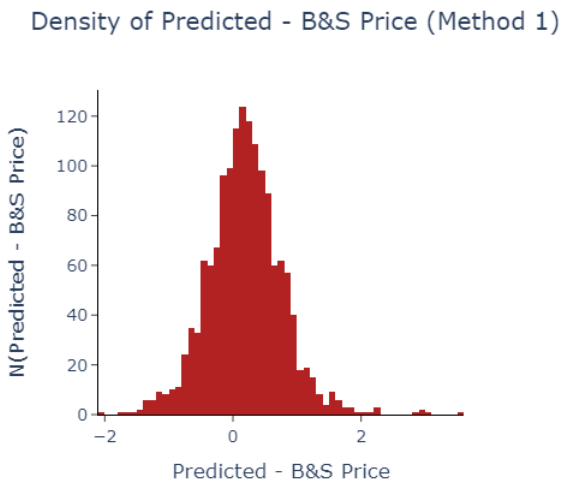


(a) Method 1.

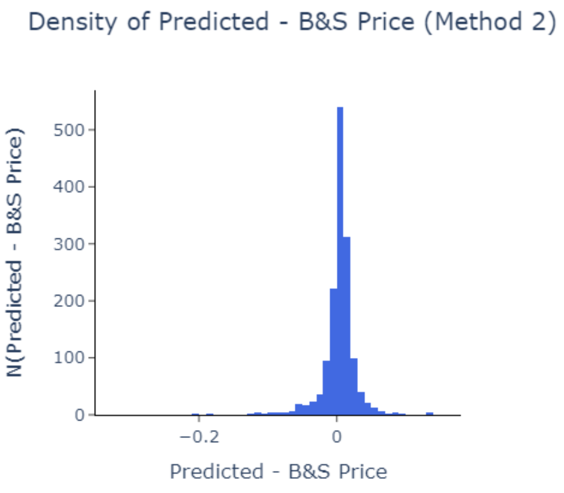


(b) Method 2.

Figure 5.3: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 3 layers and 16 nodes per layer (case 0).



(a) Method 1.



(b) Method 2.

Figure 5.4: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^4 samples, 5 layers and 64 nodes per layer (case 4).

In addition, the density of $C^D_{ANN} - C^S$ has always a mean value closer to zero and a lower variance than the density of $C^S_{ANN} - C^S$. This is stated in Table 5.1 and showed in Figure 5.3 and 5.4, where the density curve of method 2 is more symmetrical with respect to the mean, higher and steeper than the one of method 1. This means that predictions are more stable over the price range, since there are no tendencies in over-pricing or under-pricing. It is observed that simulating both prices takes always longer than evaluating only one, as in the first method. Additionally, both models take the same time in training and testing. This two reasons indicate that method 1 is more time efficient than method 2.

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
0	0.64953345	0.02428860	2.13209150	0.00816934	5.4551	4.5802
1	-0.73537450	-0.03367717	2.10095210	0.01659294	6.8981	6.3711
2	-0.25536546	-0.00428895	1.78199320	0.01566553	8.4778	8.0360
3	0.01564740	0.00094651	1.15133140	0.00403284	41.5646	38.5081
4	0.19343957	0.00292741	0.34188798	0.00093108	58.3401	59.0850
5	0.15136512	0.00207505	0.39750203	0.00088419	79.1596	81.9615
6	-0.28369760	0.01293834	5.51701500	0.08027811	4.2035	4.0202
7	-0.14649343	-0.00950860	0.39349675	0.00333944	38.1736	38.2335
8	-0.17817196	0.00370807	1.06156050	0.00412429	51.4651	56.0678
9	-0.01942846	1.2163×10^{-5}	0.00218279	9.4174×10^{-6}	4.2134	4.3020
10	0.64953345	0.02428860	2.1320915000	0.00816934	4.1731	4.1861

Table 5.1: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for each case.

Case	Mean 3	Variance 3	Time 3	MAE 3	RMSE 3
9	0.00838507	0.0049391	-	-	-
10	29.4362200	511.508420	4.2011	29.455019	37.121414

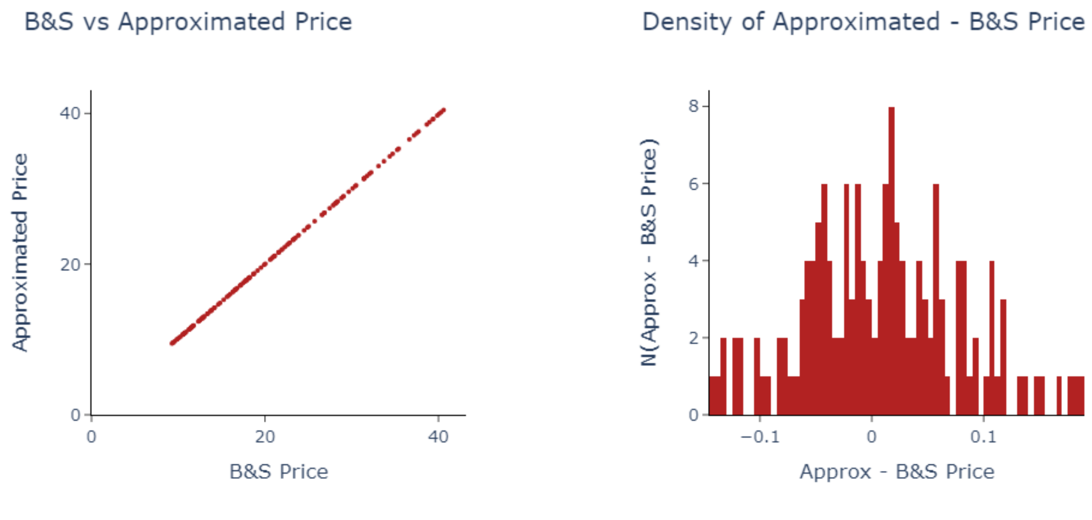
Table 5.2: Mean and variance of the density of $C^{\overline{S}} - C^S$ (case 9) and of $C_{ANN} - C^S$ (case 10), where C_{ANN} is the prediction of the price evaluated with the intrinsic formula defined in Eq.(4.1). Only in case 10 the neural network was needed and time (in seconds), MAE and RMSE are reported.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
0	1.27127680	0.05514089	1.598119	0.09359098
1	1.2767875	0.07081576	1.62533930	0.13314310
2	1.06652260	0.06178425	1.35911900	0.12523548
3	0.81012946	0.03869143	1.07311520	0.06351177
4	0.46702830	0.01677214	0.61587894	0.03065382
5	0.48383445	0.015674805	0.6483929	0.029807596
6	1.68569400	0.09924971	2.36590360	0.28362912
7	0.45113036	0.03106121	0.64417166	0.05856492
8	0.78010910	0.03053603	1.04561270	0.06432766
9	0.03897486	0.00223860	0.05059896	0.00306881
10	1.27127680	0.05514089	1.59811910	0.09359098

Table 5.3: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for each case.

Despite this, the comparison of the results with data set size of 1000 and 10000 leads to an important observation. While method 2 performs optimally in both scenarios, method 1 requires a larger set to have acceptable results. Indeed, the RMSE and MAE of C^S_{ANN} with a 1000 set are always too large for practical uses. For this reason, it can be concluded that for method 2 less simulations are necessary. This means that the computation time and cost of generating trading data is significantly reduced.
Focusing on the activation functions, ReLU performs worse than softmax, especially observing the results of the first method, where the RMSE and MSE differ more. Hence, the variation of the hyper-parameters impacts more on the results of method 1 than the ones of method 2.
Case 9 and 10 give more insight on the goodness of method 2. Table 5.2 shows the additional quantities evaluated in these scenarios. For case 9, the values listed are the mean and the variance of the density of $C^{\overline{S}} - C^S$. There are no computation time, MAE and RMSE, since no neural network has been used to evaluate these results. For case 10, the values are the mean and the variance of the density of $C_{ANN} - C^S$, where C_{ANN} is the prediction of the price evaluated with the intrinsic formula defined in Eq.(4.1). In a low volatility scenario, the Wiener-Itô Chaos expansion formula reduces to the intrinsic one. However, in the market the volatility changes greatly. Consequently, MAE and RMSE relative to $C_{ANN} - C^S$ are too high. This shows that this method is not suited for practical uses and that the additional computation time needed to evaluate $C^{\overline{S}}$ rather than

the intrinsic price is worth.



(a) C^S vs. $C^{\overline{S}}$. (b) Density of $C^{\overline{S}} - C^S$.

Figure 5.5: Comparison of the Black & Scholes price on the x-axis and the predicted prices $C^{\overline{S}}$ (left) on the y-axis and density of $C^{\overline{S}} - C^S$ (right) (case 9).

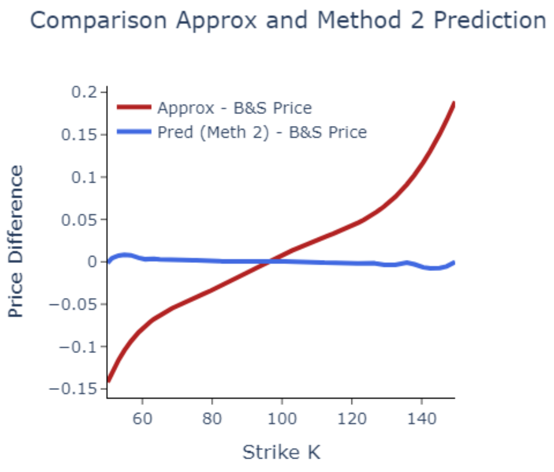
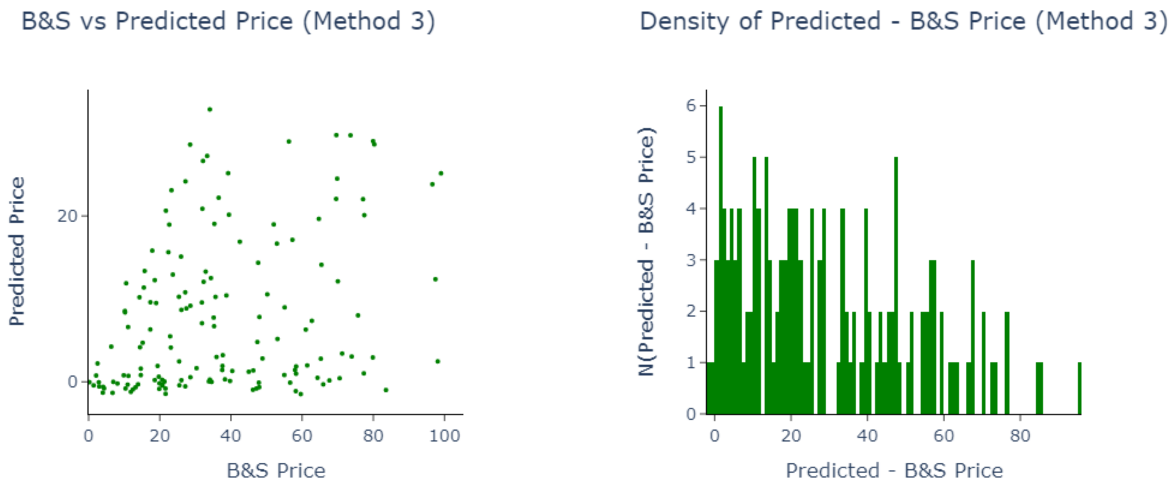


Figure 5.6: Evolution of $C^{\overline{S}} - C^S$ (red) and $C^D_{ANN} - C^S$ (blu) as the strike varies.

Concerning case 9, Figure 5.6 shows how $C^D_{ANN} - C^S$ and $C^{\overline{S}} - C^S$ behave as the strike K varies. While the difference between the asymptotic approximated price and the true B&S price varies greatly as the strike increases, the other one stays constant around zero.

This means that adopting the neural network on the difference $C^{\overline{S}} - C^S$ reduces correctly the residual term of the Wiener-Itô Chaos expansion.



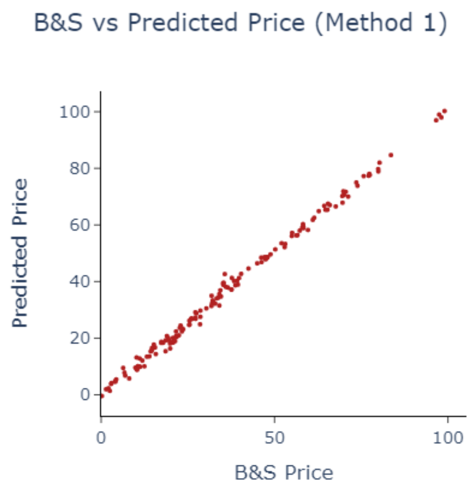
(a) C^S vs. C_{ANN}. (b) Density of $C_{ANN} - C^S$.

Figure 5.7: Comparison of the Black & Scholes price on the x-axis and the predicted prices C_{ANN} (left) on the y-axis and density of $C_{ANN} - C^S$ (right). C_{ANN} is the prediction of the price evaluated with the intrinsic formula defined in Eq.(4.1) (case 10).

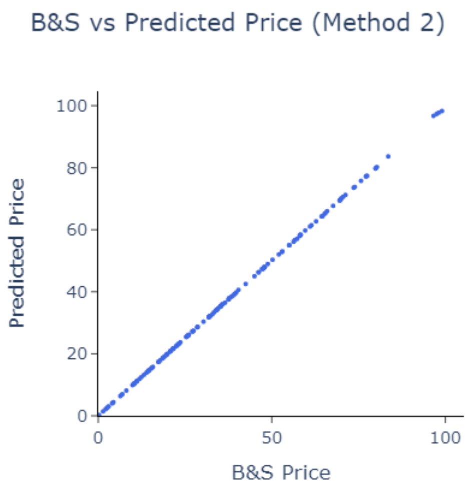
In brief, case 9 shows that applying the map $\mathscr{M}_D : \xi \mapsto D(\xi)$ improves the accuracy of the results compared to when it is not applied. Case 10 shows that using the Wiener-Itô Chaos expansion approximation rather than the intrinsic formula gives substantially better results. In conclusion, the numerical procedure of method 2 is the best among its alternatives examined.

5.1.2. Tuning the hyper-parameters: Black & Scholes

This subsection shows the results of the two methods applied to the European Call option under Black&Scholes after the hyper-parameters tuning. The hyper-parameters tuned are the number of nodes per each layer, the learning rate and the batch size. The optimizer is Adam and the number of layers and the activation functions are fixed as listed in Table 4.9.

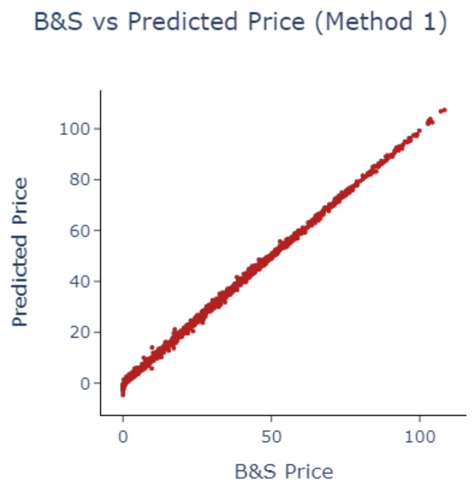


(a) Method 1.

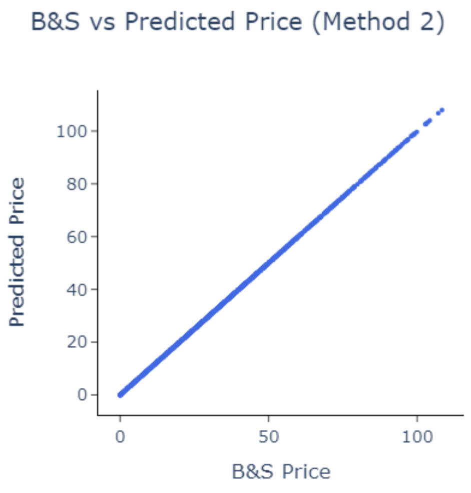


(b) Method 2.

Figure 5.8: Comparison of the Black & Scholes price on the x-axis and the predicted prices C^S_{ANN} (left) and C^D_{ANN} (right) on the y-axis with the tuned network (case 0).



(a) Method 1.

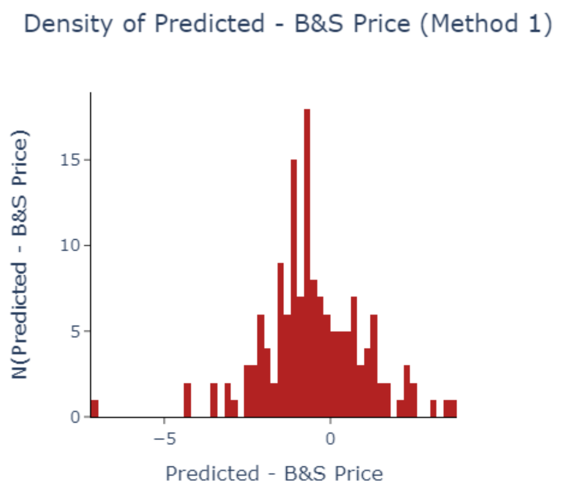


(b) Method 2.

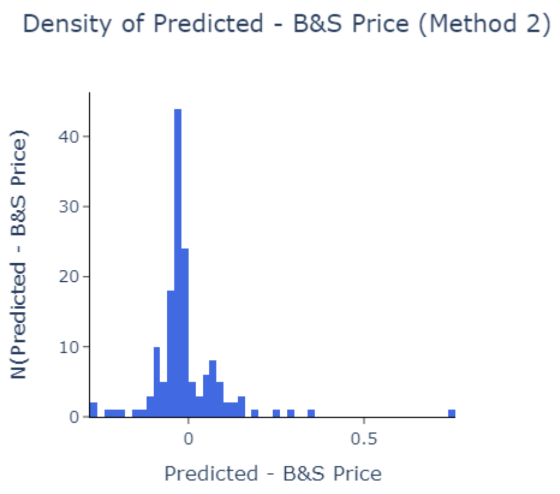
Figure 5.9: Comparison of the Black & Scholes price on the x-axis and the predicted prices C^S_{ANN} (left) and C^D_{ANN} (right) on the y-axis with the tuned network (case 4).

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
0	-0.52866906	-0.00833111	2.31357500	0.01076713	57.84	78.54
1	-0.08879303	0.02518559	2.44720580	0.00948387	73.17	101.34
2	-0.66496290	-0.00262791	2.40615900	0.00905605	101.43	128.80
3	0.03008319	-0.00382848	0.61553615	0.00123348	362.80	450.62
4	0.15955998	0.00637127	0.62561274	0.00162823	470.53	490.68
5	-0.04603379	-0.00417670	0.43935603	0.00167634	633.56	610.78
6	-0.17212744	-0.04218440	2.48194000	0.01278232	86.54	96.33
7	0.02632081	0.00191633	0.33661693	0.00187232	313.29	365.66
8	0.08615202	-0.00126915	0.80851173	0.00175655	449.26	676.71

Table 5.4: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for each case.

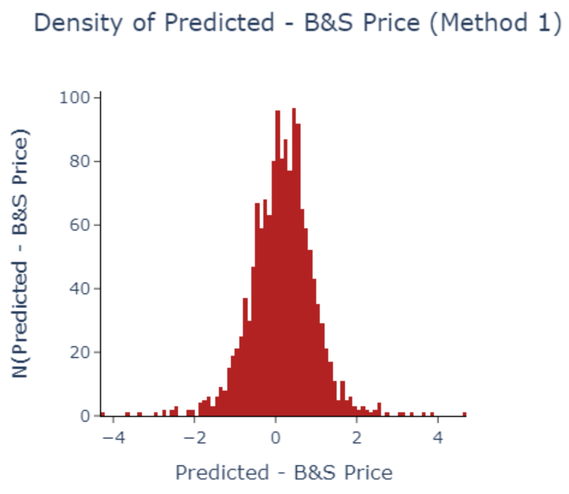


(a) Method 1.

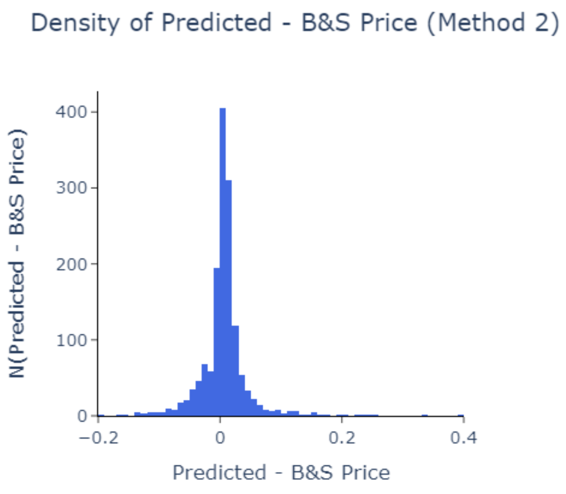


(b) Method 2.

Figure 5.10: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with the tuned network (case 0).



(a) Method 1.



(b) Method 2.

Figure 5.11: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with the tuned network (case 4).

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
0	1.25805830	0.06403027	1.61030010	0.10409872
1	1.24421630	0.04707303	1.56687260	0.10058916
2	1.41560020	0.04565541	1.68770090	0.09519958
3	0.57613150	0.02320459	0.78513770	0.03532899
4	0.60486066	0.02380361	0.80689037	0.04085128
5	0.48987060	0.02288073	0.66443600	0.04115564
6	1.23808120	0.06737506	1.58479270	0.12067246
7	0.42066652	0.02185981	0.58078370	0.04331275
8	0.65041540	0.02106299	0.90329060	0.04193042

Table 5.5: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for each case.

Table 5.6 collects the configurations of hyper-parameters selected to minimise the loss function, whose minimum value is listed in Table 5.7. Unfortunately, the comparison of the loss values and the number of iterations before hitting a plateau do not give additional insight on the methods, since the ANNs are applied to two different quantities: the call price and the difference between the call price and its approximation. Therefore, they are listed only for completeness. For what concerns the time of execution of the best trial, the one of method 1 is lower than the one in method 2. This is due to the fact that method 1 converges faster since its best configurations have higher learning rate and larger batch size.

Figures 5.8, 5.9, 5.10, 5.11 show that, even when the performance of the ANN of method 1 is maximised, is not accurate enough. In addition, observing Table 5.5, the MAE and the RMSE of method 1 are higher than the ones of method 2 for every case investigated. This means that the accuracy of method 2 is superior to the one of method 1.

Table 5.5 gives an additional insight through the comparison of the MAE and RMSE of the cases which make use of a smaller data set (0,1,2) or a larger one (3,4,5). Indeed, for method 1, the MAE and RMSE are almost halved by the increase of the train size. Conversely, for method 2 the reduction is not as relevant as in the first method, even though it is still more accurate overall. This means that method 2 is more robust with respect to the size variation of the training data and it is more computationally efficient since it does not require 10000 prices to achieve good results.

	Best configuration 1			**Best configuration 2**		
Case	**nodes per layer**	**lr**	**bs**	**nodes per layer**	**lr**	**bs**
0	64/32/64	0.0040	16	4/64/32	0.0014	2
1	64/64/8/32/16	0.0150	4	64/64/16/32/64	0.0014	8
2	64/32/32/32/8/8/16	0.0214	16	32/16/32/32/4/4/32	0.0013	2
3	64/32/64	0.0018	4	64/8/8	0.0018	4
4	64/8/64/16/32	0.0027	16	64/32/8/16/8	0.0012	2
5	64/64/4/4/32/4/16	0.0059	4	64/4/32/8/16/32/32	0.0004	8
6	64/64/64	0.0132	4	8/32/4	0.0543	16
7	64/32/64	0.0045	4	64/32/64	0.0037	4
8	64/16/16/8/16	0.0094	8	16/4/64/32/64	0.0100	16

Table 5.6: The NN hyper-parameters configurations for which the validation loss is minimised for each B&S case. The hyper-parameters tuned are the number of nodes per layer, the learning rate (lr) and the batch size (bs).

	Best performance 1				Best performance 2			
Case	**Val. loss**	**Test loss**	**Iter.**	**Time**	**Val. loss**	**Test loss**	**Iter.**	**Time**
0	0.00462060	0.00533975	5	2.1114	0.01890393	0.02314137	10	5.7468
1	0.00441384	0.00505562	5	3.9914	0.01441606	0.02160741	18	7.1841
2	0.00608327	0.00586541	6	2.1499	0.01154766	0.01935381	20	22.4751
3	0.00125447	0.00130456	4	22.1660	0.00244253	0.00254948	5	27.4094
4	0.00129539	0.00137785	4	6.6511	0.00233191	0.00340884	4	46.2563
5	0.00084346	0.00093428	7	77.5056	0.00257590	0.00345979	5	18.3401
6	0.00352215	0.00517192	7	4.4831	0.03123731	0.03109669	20	6.0257
7	0.00074587	0.00071384	4	17.4377	0.00209584	0.00383200	5	15.7532
8	0.00168070	0.00172675	4	12.1992	0.00233407	0.00359132	11	25.2049

Table 5.7: The validation loss, test loss, number of iterations and computation time of the best trial of each method for each case.

5.2. Results of the Greeks

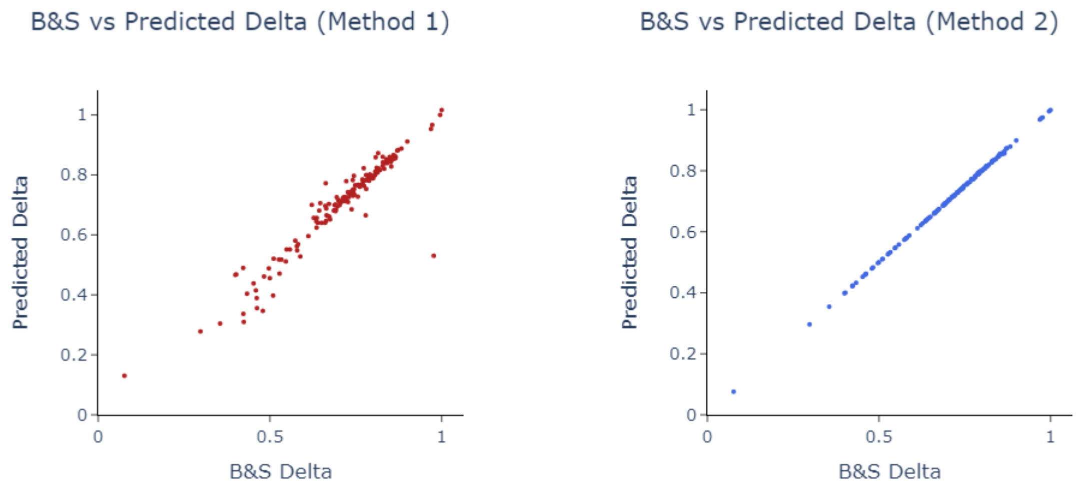
This section deals with the application of method 1 and 2 to the Greeks. In case 11 the label is the Delta of an European Call option under Black & Scholes. In case 12 and 13 the label is the corresponding Vega.
The computation time needed to simulate the Greeks for both methods is in total 0.8680 seconds.

5.2.1. Using the same hyper-parameters: Greeks

Here are listed some additional details concerning case 11, 12 and 13: for every case, the batch size is 2, the optimizer is Adam and the number of epochs is 15; for case 11 the learning rate is 0.0002, while for case 12 it is 0.0001 and for case 13 it is 0.01. The other Adam algorithm parameters are set to their default values.
In every case, method 2 is more stable than method 1: in Table 5.8 the mean deviates less from zero and the variance is lower. This difference is particularly relevant in case 13, in which the Vega is analysed in *at-the-money* scenario. Indeed, when it occurs, the Vega assumes a bell-shape [26] and for this reason, the price variations with respect to ϵ are rapid, especially with small volatility. So method 2 shows fast convergence and stable results even in this scenario, while method 1 converges poorly.

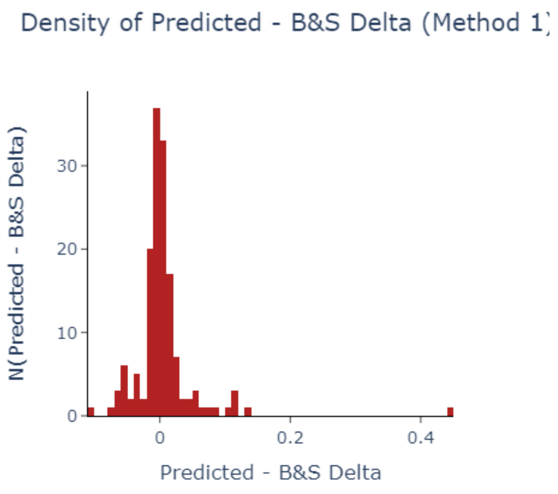
Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
11	0.00420489	7.7100×10^{-5}	0.00235573	2.3611×10^{-6}	6.8628	6.6115
12	0.66665140	-0.21314749	44.09313000	0.22350869	6.2195	7.1780
13	-0.27505130	-1.4157×10^{-7}	0.59884006	6.3592×10^{-11}	4.5443	4.4426

Table 5.8: Mean and variance of the densities of the difference of the predicted Greek and the Black & Scholes Greek and time to run the neural networks (in seconds) for each case and each method.

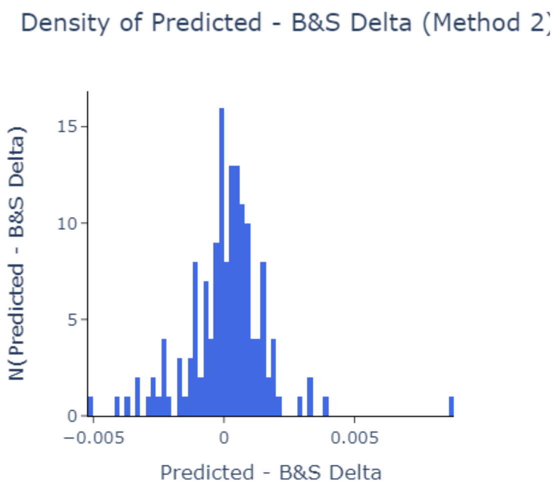


(a) Method 1. (b) Method 2.

Figure 5.12: Comparison of the Black & Scholes Delta on the x-axis and the predicted $Delta^S_{ANN}$ (left) and $Delta^D_{ANN}$ (right) on the y-axis with 10^3 samples, 5 layers and 64 nodes per layers (case 11).



(a) Method 1.



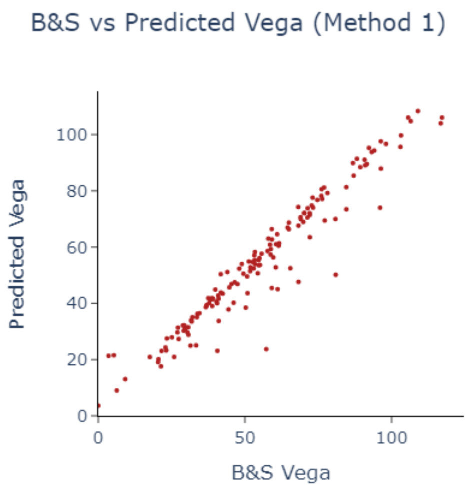
(b) Method 2.

Figure 5.13: Comparison of the densities of $Delta^S_{ANN} - Delta^S$ (left) and $Delta^D_{ANN} - Delta^S$ (right) with 10^3 samples, 5 layers and 64 nodes per layer (case 11).

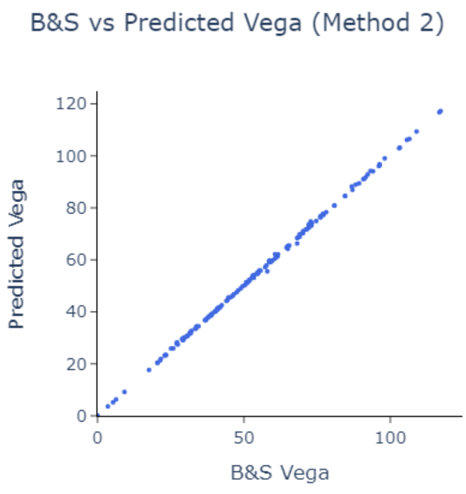
Moreover, as shown in Table 5.9, method 2 produces more accurate estimates. Especially for the Vega in a *at-the-money* scenario, the MAE and the RMSE of method 2 are extremely low.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
11	0.02429887	0.00105230	0.04987107	0.00153852
12	4.07167630	0.31870645	6.67364650	0.51859474
13	0.46027660	5.2967×10^{-6}	0.82127540	9.2651×10^{-5}

Table 5.9: RMSE and MAE of the predicted Greek for each case and each method.

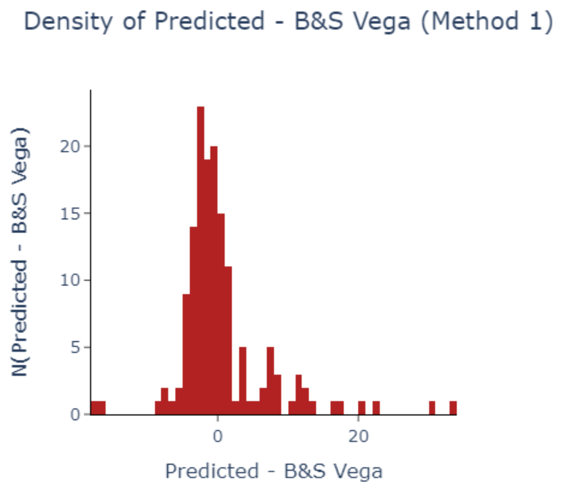


(a) Method 1.

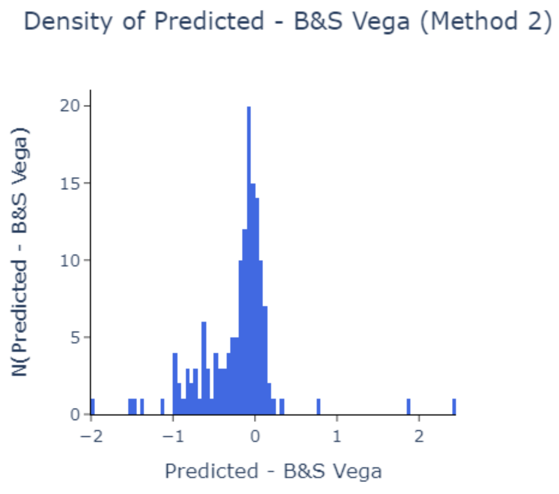


(b) Method 2.

Figure 5.14: Comparison of the Black & Scholes Vega on the x-axis and the predicted $Vega^S_{ANN}$ (left) and $Vega^D_{ANN}$ (right) on the y-axis with 10^3 samples, 5 layers and 64 nodes per layers (case 12).

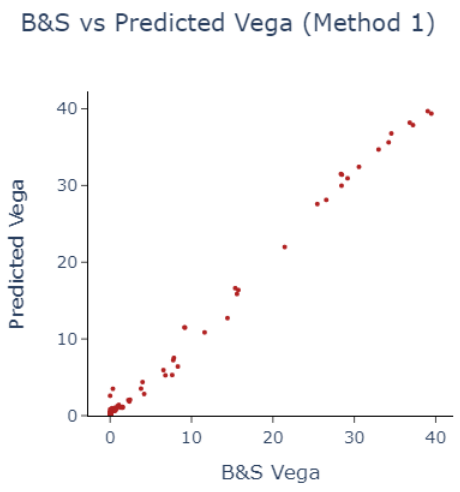


(a) Method 1.

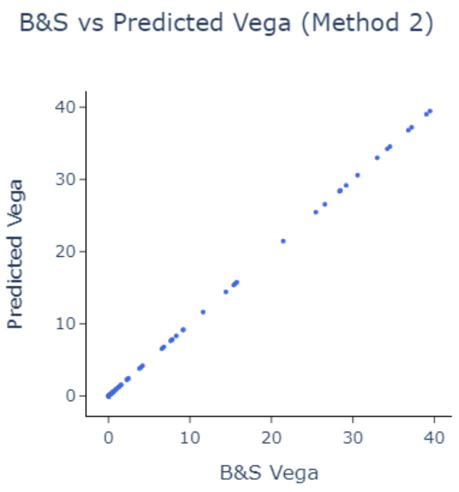


(b) Method 2.

Figure 5.15: Comparison of the densities of $Vega^S_{ANN} - Vega^S$ (left) and $Vega^D_{ANN} - Vega^S$ (right) with 10^3 samples, 5 layers and 64 nodes per layer (case 12).

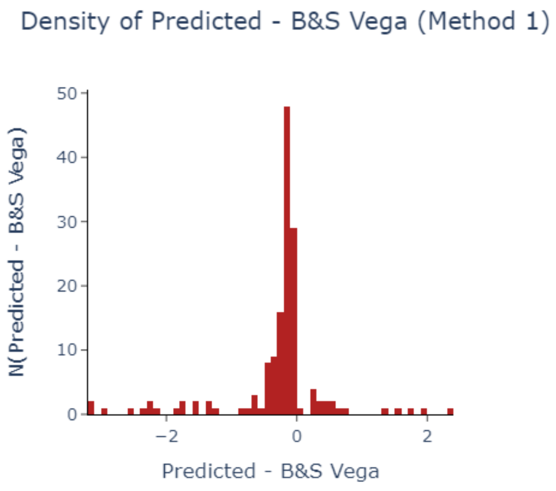


(a) Method 1.

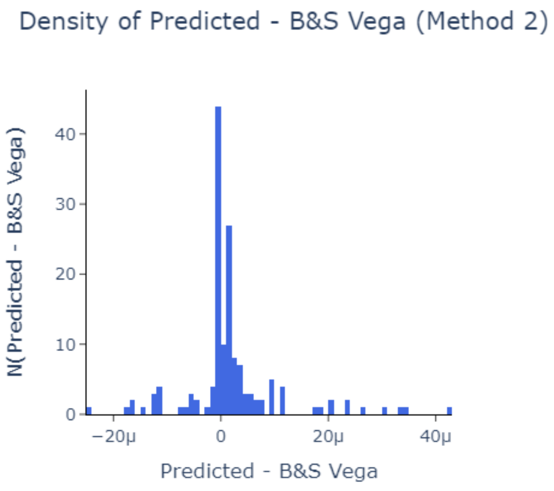


(b) Method 2.

Figure 5.16: Comparison of the Black & Scholes Vega on the x-axis and the predicted $Vega^S_{ANN}$ (left) and $Vega^D_{ANN}$ (right) on the y-axis with 10^3 samples, 3 layers and 16 nodes per layers in a *at-the-money* scenario (case 13).



(a) Method 1.



(b) Method 2.

Figure 5.17: Comparison of the densities of $Vega^S_{ANN} - Vega^S$ (left) and $Vega^D_{ANN} - Vega^S$ (right) with 10^3 samples, 3 layers and 64 nodes per layer in a *at-the-money* scenario (case 13).

5.2.2. Tuning the hyper-parameters: Greeks

Here the Greeks results after the hyper-parameters tuning are discussed.
Table 5.12 shows the configurations attained by the optimized neural network, while Table 5.13 lists its validation and test loss values. Moreover, Table 5.10 shows the mean and the variance of the prediction errors. Especially for case 12, the variance of method 1 is consistently reduced by the hyper-parameters optimization phase.

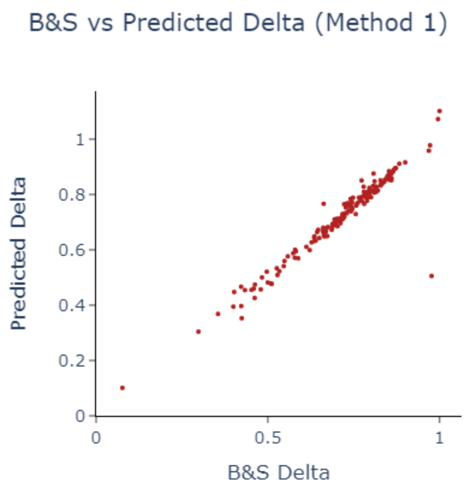
Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
11	-0.00618268	-0.00059838	0.00204158	1.5244×10^{-6}	106.18	103.70
12	1.906002	-0.05679214	15.1198600	0.09738996	98.21	124.10
13	-0.05719818	-9.1276×10^{-7}	0.81977785	1.2329×10^{-10}	109.70	96.00

Table 5.10: Mean and variance of the densities of the difference of the predicted Greek and the Black & Scholes Greek and time to run the neural networks (in seconds) for each case and each method.

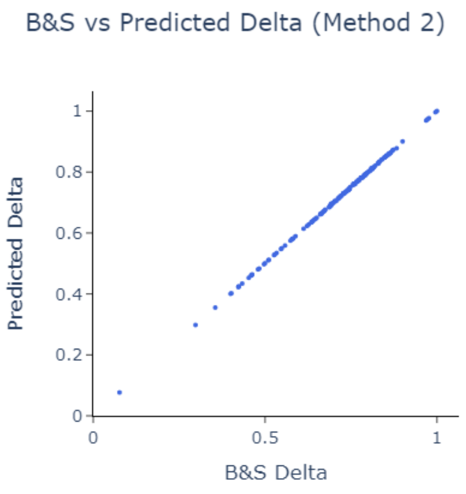
For what concerns the measures of errors, Table 5.11 indicates that method 1 is less accurate than method 2, since both its MAE and RMSE are higher than the ones of method 2. This holds especially for case 13, where the Vega is measured in a *at-the-money* scenario. This is due to the fact that method 2 does not face the exploding gradient problem, since the partial derivation is operated on the difference and not directly on the price, which is bell-shaped.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
11	0.02025129	0.00109659	0.04560494	0.00137203
12	3.0285408	0.2090156	4.330439	0.31719914
13	0.55224700	6.6031×10^{-6}	0.90722070	1.1141×10^{-5}

Table 5.11: RMSE and MAE of the predicted Greek for each case and each method.

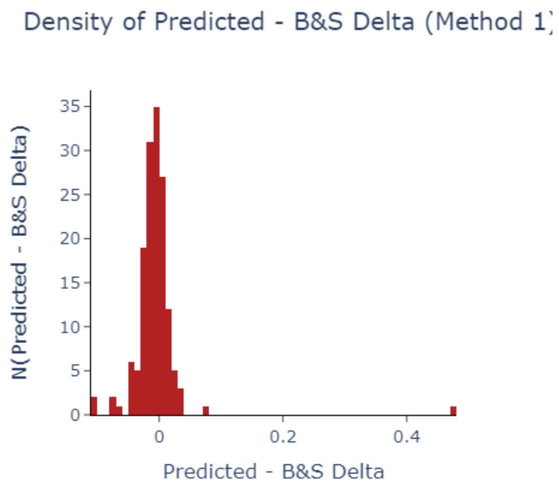


(a) Method 1.

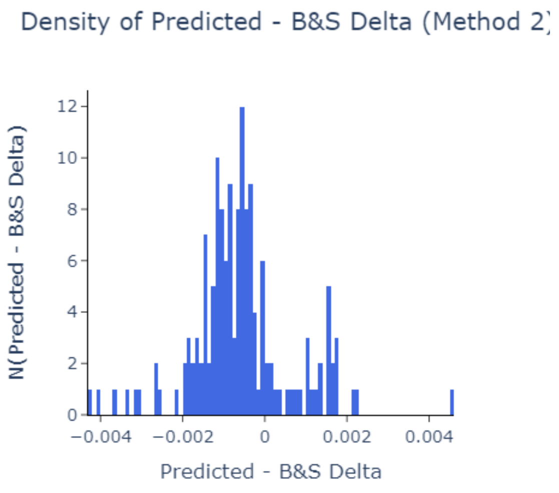


(b) Method 2.

Figure 5.18: Comparison of the Black & Scholes Delta on the x-axis and the predicted $Delta^S_{ANN}$ (left) and $Delta^D_{ANN}$ (right) on the y-axis with the tuned network (case 11).

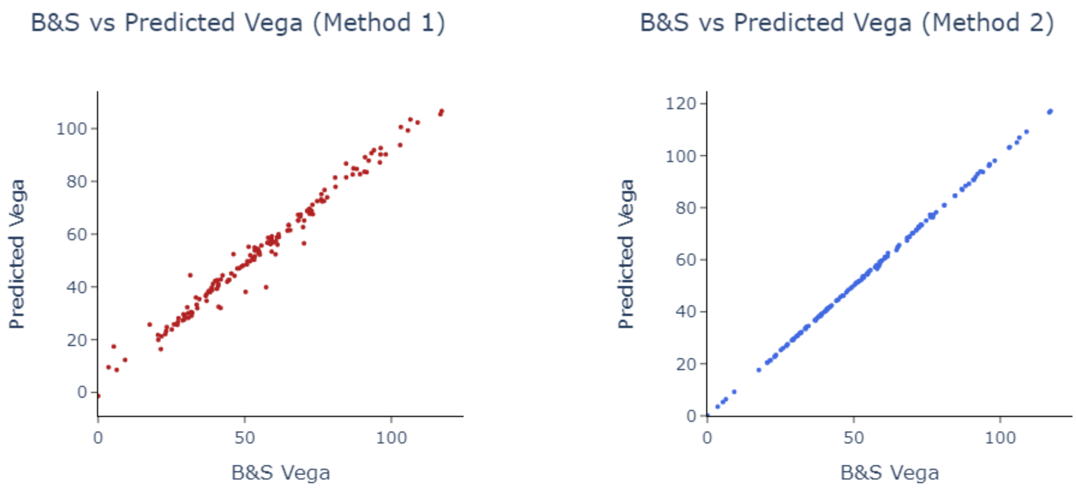


(a) Method 1.



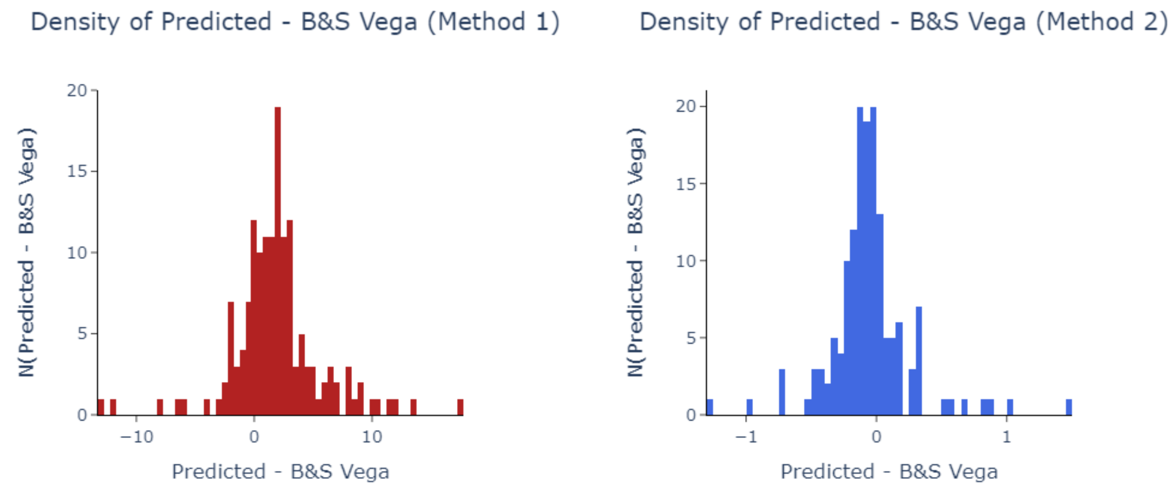
(b) Method 2.

Figure 5.19: Comparison of the densities of $Delta^S_{ANN} - Delta^S$ (left) and $Delta^D_{ANN} - Delta^S$ (right) wwith the tuned network (case 11).



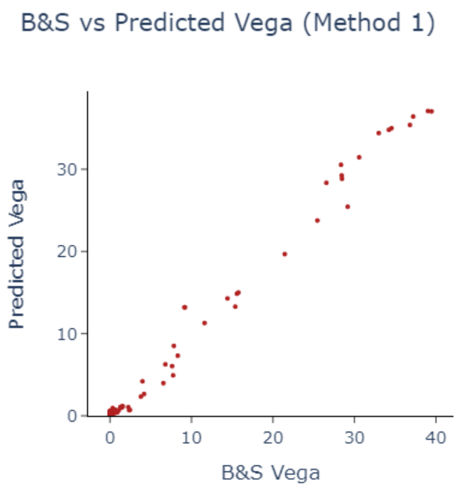
(a) Method 1. (b) Method 2.

Figure 5.20: Comparison of the Black & Scholes Vega on the x-axis and the predicted $Vega^S_{ANN}$ (left) and $Vega^D_{ANN}$ (right) on the y-axis with the tuned network (case 12).

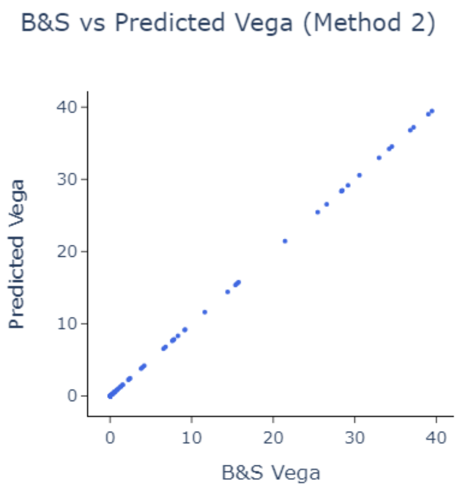


(a) Method 1. (b) Method 2.

Figure 5.21: Comparison of the densities of $Vega^S_{ANN} - Vega^S$ (left) and $Vega^D_{ANN} - Vega^S$ (right) with the tuned network (case 12).

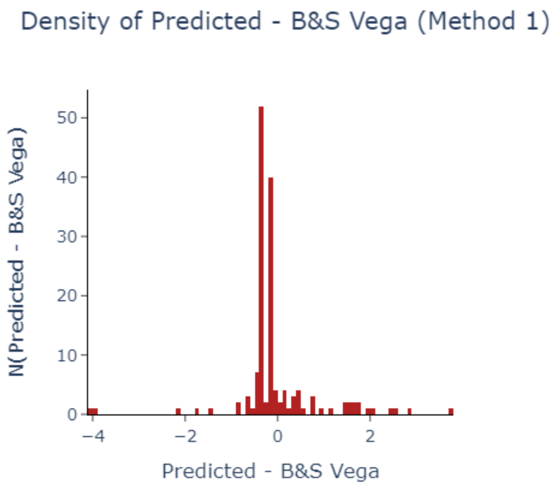


(a) Method 1.

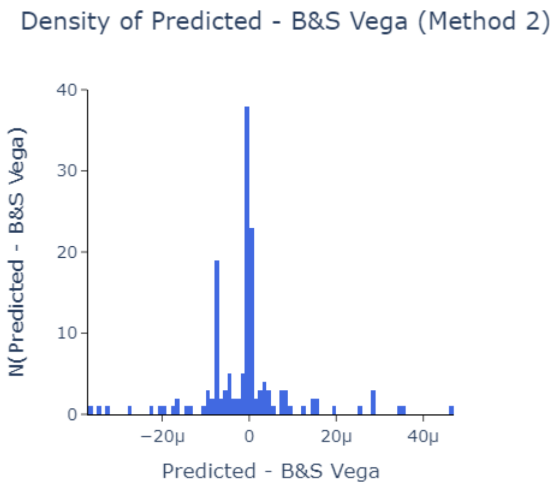


(b) Method 2.

Figure 5.22: Comparison of the Black & Scholes Vega on the x-axis and the predicted $Vega^{S}_{ANN}$ (left) and $Vega^{D}_{ANN}$ (right) on the y-axis with 10^3 samples, 3 layers and 16 nodes per layers in a *at-the-money* scenario (case 13).



(a) Method 1.



(b) Method 2.

Figure 5.23: Comparison of the densities of $Vega^{S}_{ANN} - Vega^{S}$ (left) and $Vega^{D}_{ANN} - Vega^{S}$ (right) with the tuned network in a *at-the-money* scenario (case 13).

	Best configuration 1			**Best configuration 2**		
Case	**nodes per layer**	**lr**	**bs**	**nodes per layer**	**lr**	**bs**
11	64/32/4/4/4	0.0098	16	4/16/4/32/64	0.0051	4
12	16/64/4/8/64	0.0061	8	32/8/64/32/32	0.0010	2
13	32/16/8	0.0769	8	64/4/4	0.0292	8

Table 5.12: The NN hyper-parameters configurations for which the validation loss is minimised for each B&S Greek case. The hyper-parameters tuned are the number of nodes per layer, the learning rate (lr) and the batch size (bs).

	Best performance 1				**Best performance 2**			
Case	**Val. loss**	**Test loss**	**Iter.**	**Time**	**Val. loss**	**Test loss**	**Iter.**	**Time**
11	0.05271084	0.08757344	11	3.5498	0.00486848	0.00548440	6	4.5190
12	0.08365190	0.03262035	17	4.5190	0.00802126	0.01049336	16	24.3003
13	0.03819393	0.00839510	20	10.2698	0.46160635	0.49732201	20	21.9552

Table 5.13: The validation loss, test loss, number of iterations and computation time of the best trial of each method for each case.

5.3. Results of the Up-and-In barrier under Heston

In this section, the cases 14 and 15 are examined. Both methods are applied to the Up-and-In barrier price under the Heston model. The Monte Carlo simulation of the prices took 208558.5848 seconds, while the evaluation of the asymptotic expansion price took an additional 193494 seconds approximately. Compared to the previous cases, these price simulations and the calculation of their approximations are very computationally expensive.

5.3.1. Using the same hyper-parameters: Heston

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
14	-0.33899918	-0.08203293	6.12968870	2.17119960	5.7656	5.7487
15	-0.97843890	-0.14020070	7.04063460	1.95420720	10.6770	10.5278

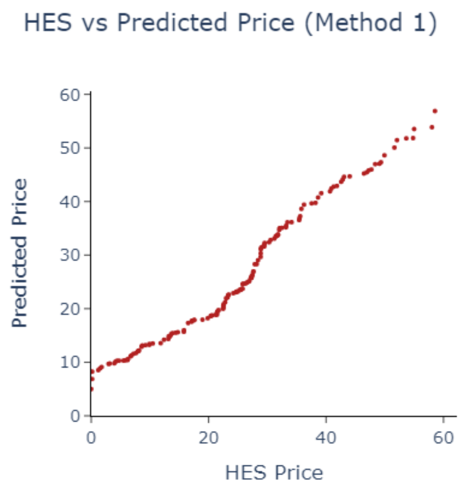
Table 5.14: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for each case.

Regarding the neural networks, some hyper-parameters used are the same for both cases: the batch size is 2, the optimizer is Adam and the number of epochs is 20. The Adam algorithm parameters are set to their default values, with the exception of the learning rate: for case 14 is 0.00005, while for case 15 it is 0.00002. Another difference between the two cases is the number of layers used.
Observing Figure 5.24, we notice that in method 1 low prices (<20) are slightly over-priced, while medium prices (between 20 and 30) are under-priced. This phenomenon does not occur for method 2, whose predictions are more stable over the whole price range.

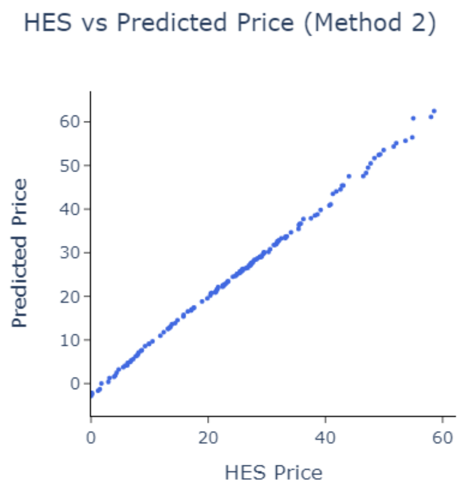
Case	MAE 1	MAE 2	RMSE 1	RMSE 2
14	1.99540910	0.98431855	2.49892140	1.47578070
15	2.32346500	0.92629355	2.82806940	1.40494250

Table 5.15: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for each case.

This model confirms that method 2 is more accurate than method 1. Table 5.15 reports lower values of RMSE and MAE for method 2.
Table 5.14 shows that the convergence is faster in method 2 and that the number of layers does not impact much on the results. The huge additional time to compute the approximated price $C^{\overline{S}}$ is the great downside of method 2. However, the alternative method 1 is not accurate enough for practical uses. In conclusion, method 2 is still preferable than method 1.

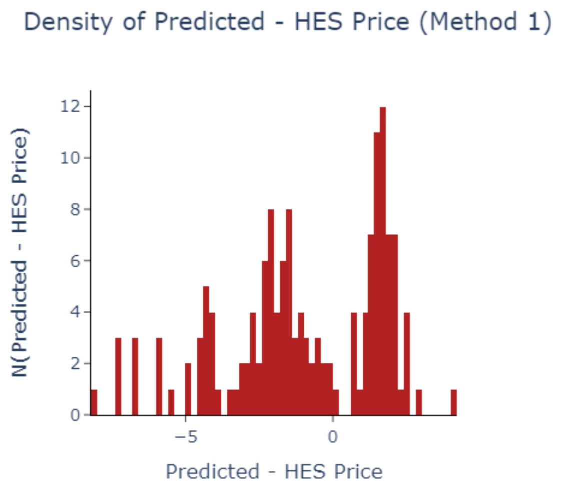


(a) Method 1.

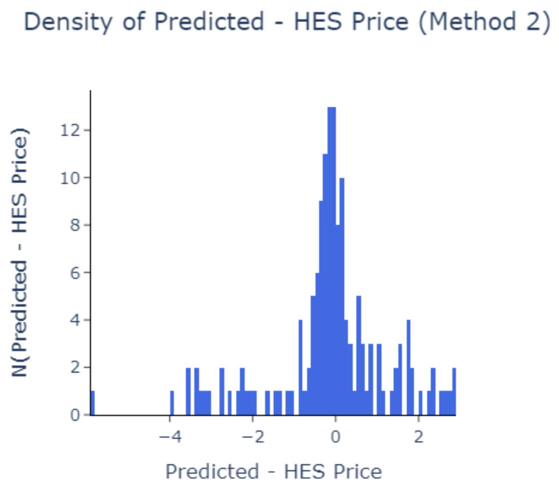


(b) Method 2.

Figure 5.24: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 7 layers and 64 nodes per layer (case 15).



(a) Method 1.



(b) Method 2.

Figure 5.25: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 7 layers and 64 nodes per layer (case 15).

5.3.2. Tuning the hyper-parameters: Heston

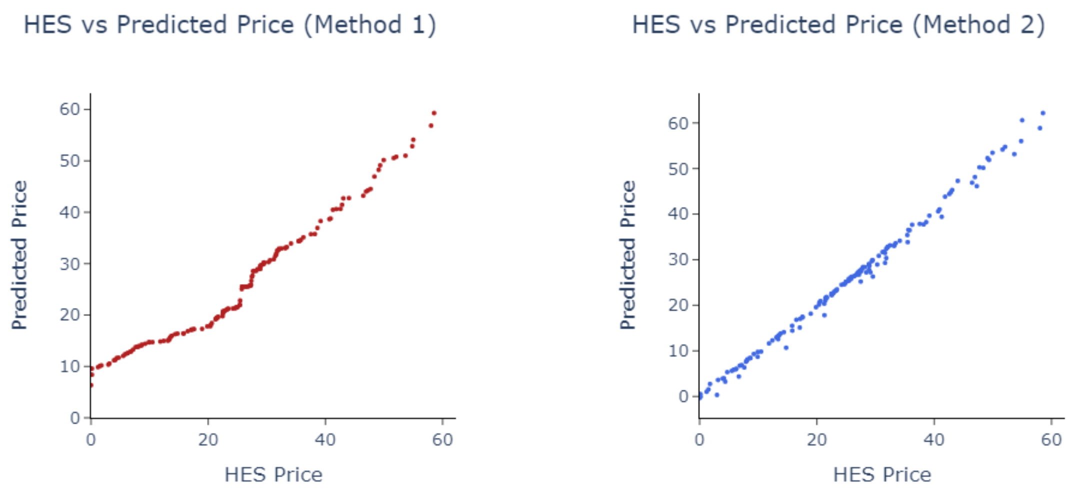
This subsections examines the results of the Up-and-In barrier option under Heston, after the hyper-parameters optimization. Method 2 is more accurate than method 1, since its

MAE and RMSE in Table 5.17 are lower than the ones of method 1. Moreover, if the measures of errors are compared between cases 14 and 15, their difference is much more relevant for method 1 than method 2. This shows that method two has a greater stability with respect to the choice of number of layers of the ANN.

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
14	-1.2137996	-0.07696503	5.0599985	1.6545022	82.40	96.00
15	-0.6335439	-0.18436617	10.556029	1.3943702	143.90	149.30

Table 5.16: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for each case.

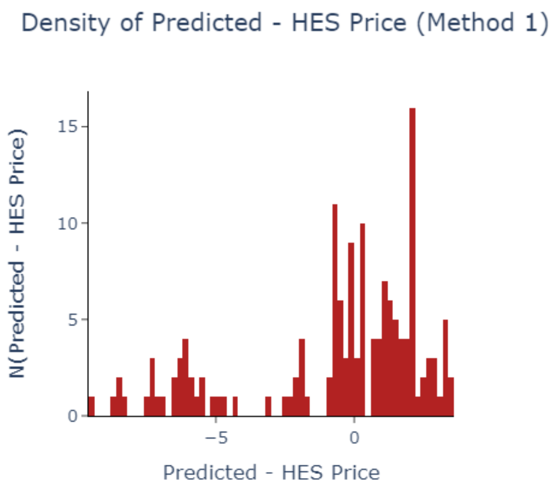
Tables 5.19 and 5.18 list the best performance measures and the hyper-parameters best performing sets, respectively.
Figures 5.26a and 5.27a show clearly that predicted prices are shifted with respect to their true prices in method 1. In particular, this phenomenon occurs for low prices (lower than 20). Method 2 is more stable in predicting the prices independently of the price range.



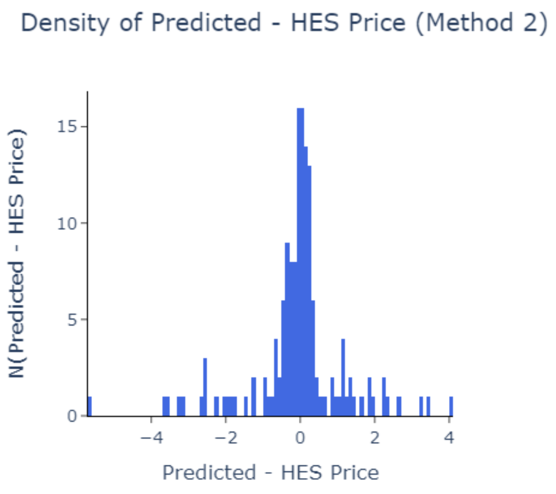
(a) Method 1.

(b) Method 2.

Figure 5.26: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with the tuned network (case 15).



(a) Method 1.



(b) Method 2.

Figure 5.27: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with with the tuned network (case 15).

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
14	1.73985760	0.76840090	2.55603360	1.28857500
15	2.38384600	0.74057525	3.31019760	1.24215170

Table 5.17: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for each case.

	Best configuration 1			Best configuration 2		
Case	nodes per layer	lr	bs	nodes per layer	lr	bs
14	8/16/64	0.0197	8	32/32/64	0.0021	16
15	8/32/32/8/16/8	0.0130	8	16/32/32/32/32/64/16	0.0011	16

Table 5.18: The NN hyper-parameters configurations for which the validation loss is minimised for each Heston case. The hyper-parameters tuned are the number of nodes per layer, the learning rate (lr) and the batch size (bs).

	Best performance 1				Best performance 2			
Case	**Val. loss**	**Test loss**	**Iter.**	**Time**	**Val. loss**	**Test loss**	**Iter.**	**Time**
14	0.16490477	0.18632613	9	4.3440	0.12472408	0.45735435	14	6.8924
15	0.16304855	0.18545772	16	9.5927	0.14401061	0.47959751	11	4.8523

Table 5.19: The validation loss, test loss, number of iterations and computation time of the best trial of each method for each case.

5.4. Results of the Down-and-In barrier under CEV

Case 16 is evaluated in this section. The Brownian Bridge Monte Carlo took 11907 seconds, while the approximation pricing took 108 seconds *circa*. The latter computation is quite fast since the short interest rate has been fixed to zero and this speeds up the procedure compared to when r is not null.

5.4.1. Using the same hyper-parameters: CEV

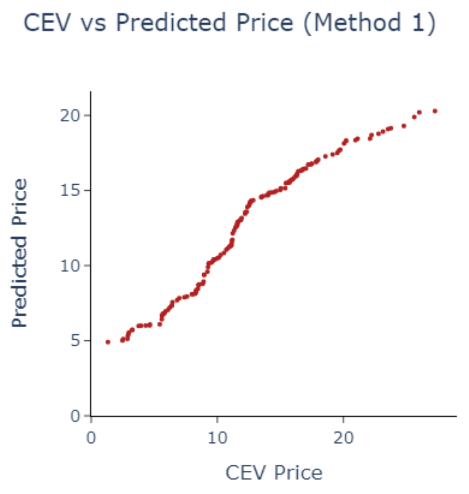
The hyper-parameters are set as: the batch size is 2, the optimizer is Adam, the learning rate is 0.001 and the number of epochs is 20.

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
16	-0.11889211	0.02371265	2.91692500	0.06843765	5.3125	5.3757

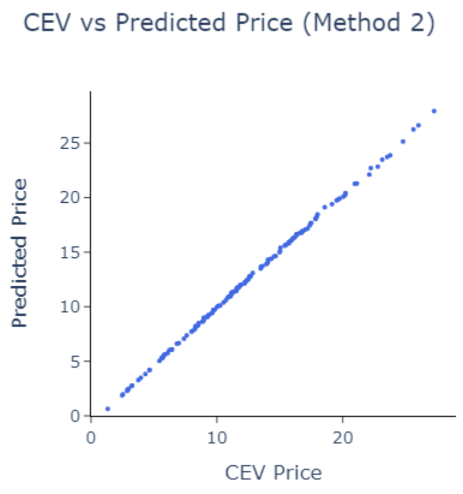
Table 5.20: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for case 16.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
16	1.18054200	0.20661426	1.71203400	0.26267840

Table 5.21: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for case 16.

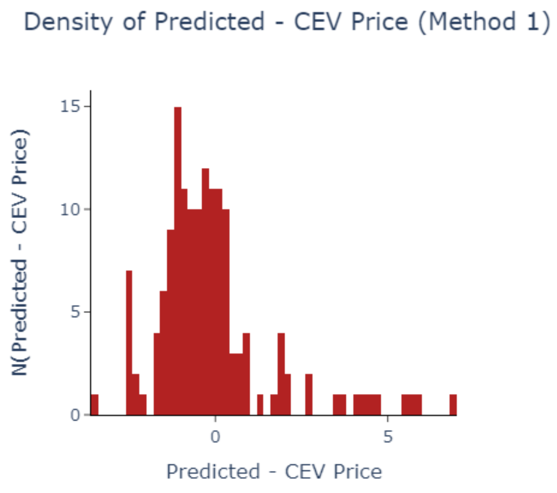


(a) Method 1.

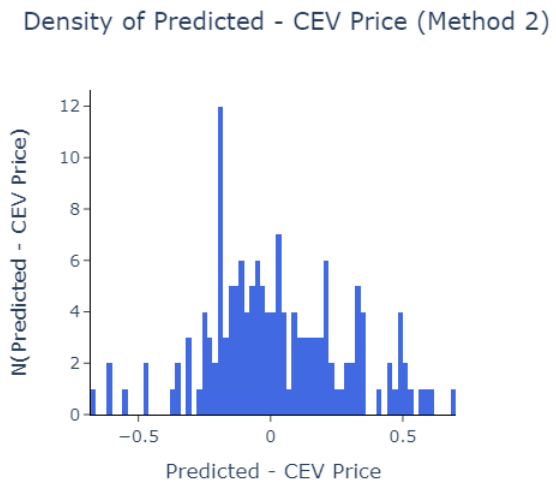


(b) Method 2.

Figure 5.28: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 3 layers and 16 nodes per layer (case 16).



(a) Method 1.



(b) Method 2.

Figure 5.29: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 3 layers and 16 nodes per layer (case 16).

Clearly method 1 is not accurate enough to price the Down-and-In barrier. Indeed, Figures 5.28 and 5.29 show that method 2 is evidently superior, since it predicts correctly the option price while method 1 is often not reliable at all. Especially in Figure 5.28, it is shown that method 1 tends to over-price lower prices and under-price higher prices. Method 2 does not have this tendency. Tables 5.20 and 5.21 confirm these results, listing un-centred mean and high error variance and RMSE and MAE for method 1.

5.4.2. Tuning the hyper-parameters: CEV

Here the Down-and-In prices under the CEV model after the hyper-parameters tuning are examined.

The resulting MAE and RMSE in Table 5.23 indicate that method 2 is more accurate than method 1, since its measures of error are lower.

In this case, Figures 5.30a and 5.31a illustrate that in method 1 low (<5) and high (>20) predicted prices are shifted with respect to their true prices. This event does not occur in method 2, which indicates that this method is more stable than method 1 in every price range.

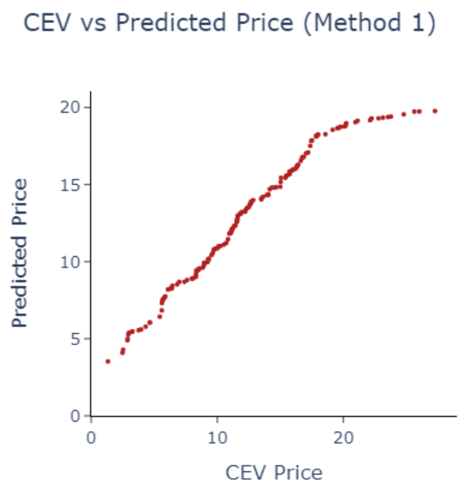
Tables 5.24 and 5.25 state the hyper-parameters chosen and the loss values attained with that particular configuration.

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
16	-0.40716386	0.01347228	2.61642340	0.05832489	105.88	97.20

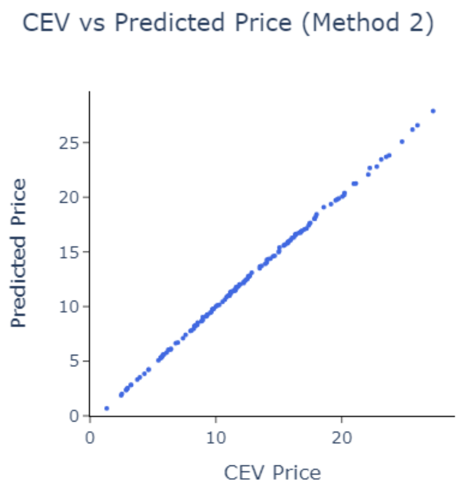
Table 5.22: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for case 16.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
16	1.19669310	0.18829569	1.66799450	0.24188094

Table 5.23: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for case 16.

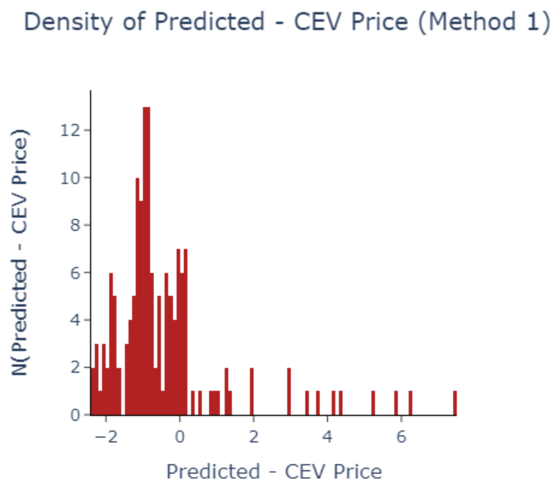


(a) Method 1.

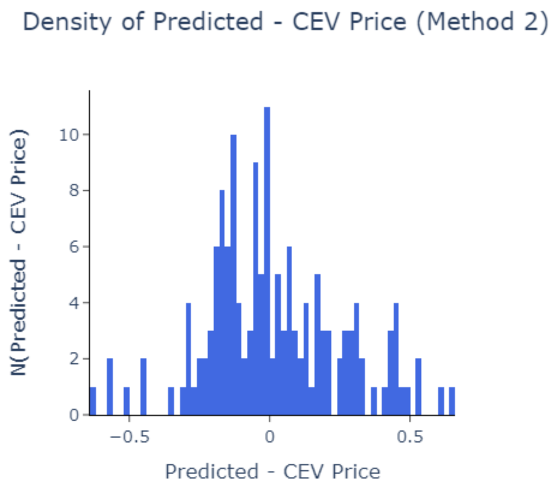


(b) Method 2.

Figure 5.30: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with the tuned network (case 16).



(a) Method 1.



(b) Method 2.

Figure 5.31: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with the tuned network (case 16).

	Best configuration 1			**Best configuration 2**		
Case	**nodes per layer**	**lr**	**bs**	**nodes per layer**	**lr**	**bs**
16	8/8/4	0.0052	4	8/16/64	0.0124	4

Table 5.24: The NN hyper-parameters configurations for which the validation loss is minimised for each the CEV case. The hyper-parameters tuned are the number of nodes per layer, the learning rate (lr) and the batch size (bs).

	Best performance 1				**Best performance 2**			
Case	**Val. loss**	**Test loss**	**Iter.**	**Time**	**Val. loss**	**Test loss**	**Iter.**	**Time**
16	0.55564883	0.52166334	9	5.8292	0.32310065	0.27133182	9	6.3661

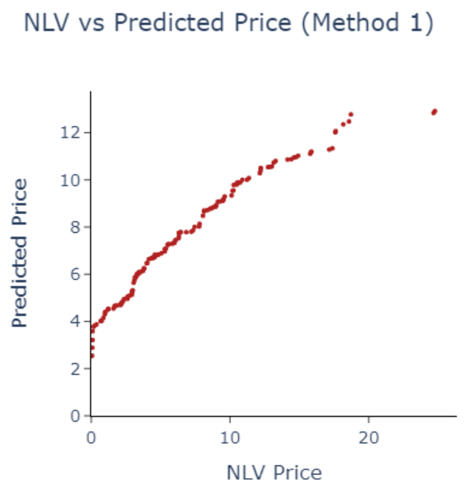
Table 5.25: The validation loss, test loss, number of iterations and computation time of the best trial of each method for case 16.

5.5. Results of the Down-and-In barrier under the non linear volatility model

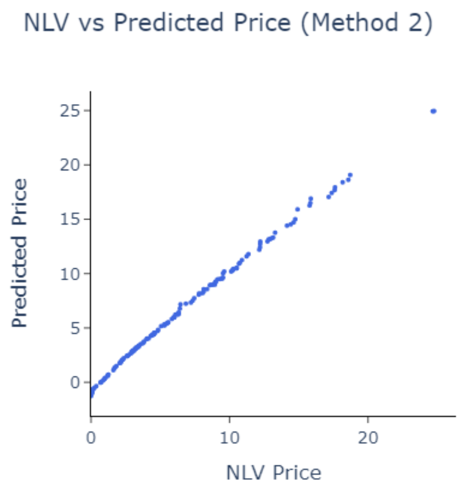
This section examines case 16, so the Down-and-In barrier under the non linear volatility model defined by Eqs. (1.10) and (1.11). The Brownian Bridge Monte Carlo took around 15018 seconds, while the computation of the approximation took only 127 seconds. Similarly to the previous case 17, the evaluation of the approximated price is sped up by setting r as null.

5.5.1. Using the same hyper-parameters: NLV

The hyper-parameters are selected as follows: the batch size is 2, the optimizer is Adam, the number of epochs is 10 and the learning rate is 0.001.
This case confirms, once again, that method 1 performs much worse than method 2. Figure 5.32 shows that the actual prices are predicted incorrectly in method 1. Figure 5.33 and Table 5.26 show that the density of the difference between the predicted price and the true one of method 1 is not centred in zero and has a high variance. Conversely, method 2 is more stable and is much more accurate, as the RMSE and MAE in Table 5.27 are much lower. For these reasons, the second method is preferable to the first one.

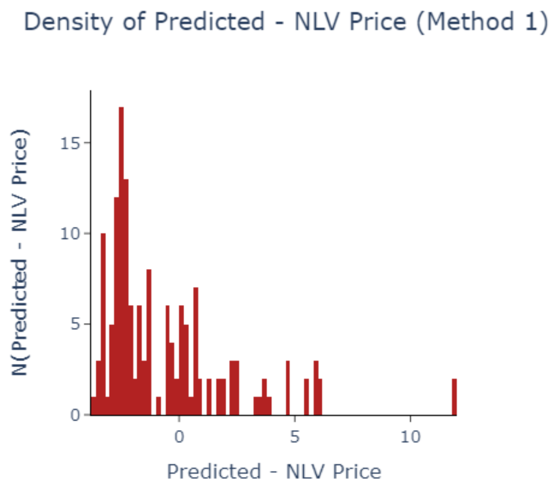


(a) Method 1.

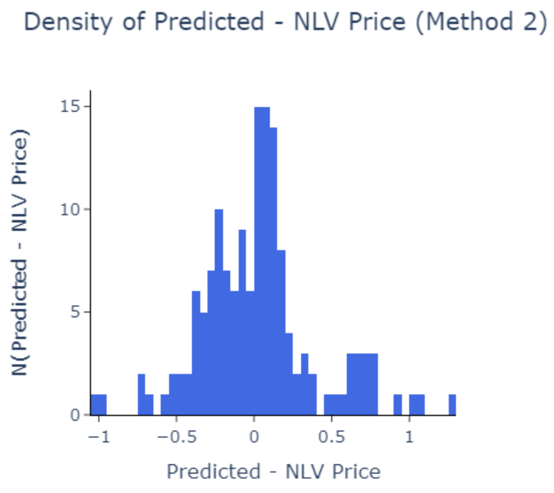


(b) Method 2.

Figure 5.32: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 3 layers and 16 nodes per layer (case 17).



(a) Method 1.



(b) Method 2.

Figure 5.33: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with 10^3 samples, 3 layers and 16 nodes per layer (case 17).

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
17	-0.57579404	0.02639847	8.02863800	0.13829833	5.5129	6.5108

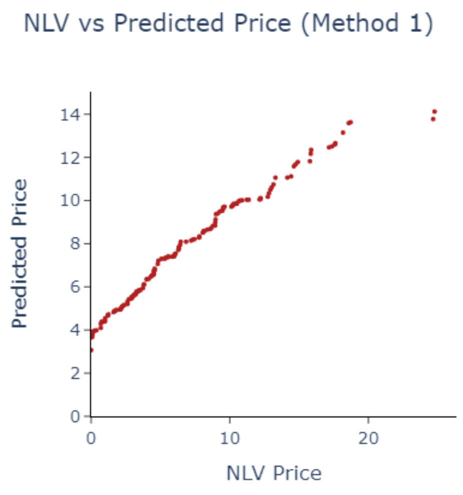
Table 5.26: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for case 17.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
17	2.31014540	0.26924598	2.89139700	0.37282062

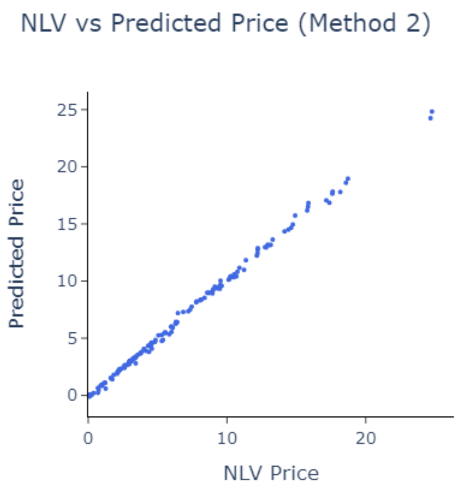
Table 5.27: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for case 17.

5.5.2. Tuning the hyper-parameters: NLV

This subsection reports the results of the Down-and-In barrier option under the non-linear volatility model.
Tables 5.31 and 5.30 list the best performance measures and the hyper-parameters best performing sets, respectively.

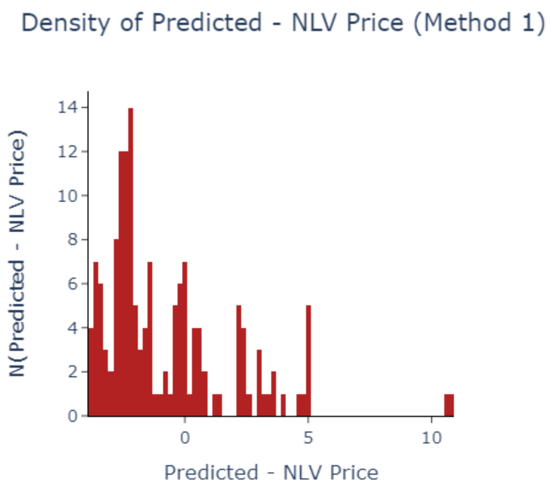


(a) Method 1.

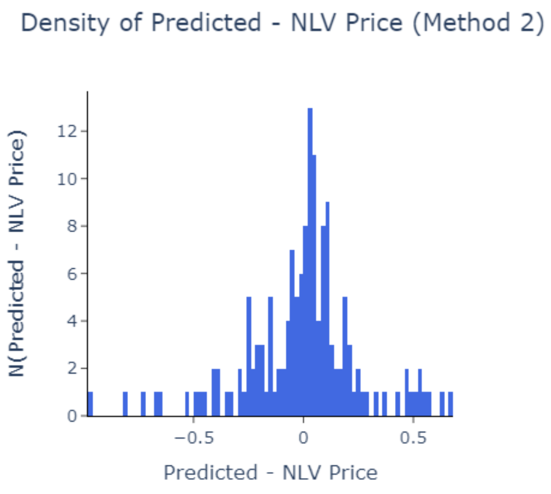


(b) Method 2.

Figure 5.34: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right)with the tuned network (case 17).



(a) Method 1.



(b) Method 2.

Figure 5.35: Comparison of the densities of $C^S_{ANN} - C^S$ (left) and $C^D_{ANN} - C^S$ (right) with the tuned network (case 17).

Case	Mean 1	Mean 2	Variance 1	Variance 2	Time 1	Time 2
17	-0.76990753	-0.00489157	7.15549660	0.06924574	96.40	102.9

Table 5.28: Mean and variance of the densities of $C^S_{ANN} - C^S$ (method 1) and $C^D_{ANN} - C^S$ (method 2) and time to run the neural networks (in seconds) for case 17.

This case confirms that method 1 is less accurate than method 2. Table 5.29 indicates that MAE and RMSE are lower in the second method rather than the first.
Figures 5.34a and 5.35a show that all predicted prices are shifted with respect to true prices. This behaviour expresses that method 1 is unfeasible for practical applications. This does not occur in Figure 5.34b and 5.35b. The small shift showed in the un-tuned case in Figure 5.32b is corrected with the hyper-parameters optimization.

Case	MAE 1	MAE 2	RMSE 1	RMSE 2
17	2.29631040	0.18381032	2.78356840	0.26319130

Table 5.29: RMSE and MAE of C^S_{ANN} (method 1) and C^D_{ANN} (method 2) for case 17.

	Best configuration 1			**Best configuration 2**		
Case	**nodes per layer**	**lr**	**bs**	**nodes per layer**	**lr**	**bs**
17	4/8/32	0.0242	16	4/16/16	0.0026	16

Table 5.30: The NN hyper-parameters configurations for which the validation loss is minimised for the NLV case. The hyper-parameters tuned are the number of nodes per layer, the learning rate (lr) and the batch size (bs).

	Best performance 1				**Best performance 2**			
Case	**Val. loss**	**Test loss**	**Iter.**	**Time**	**Val. loss**	**Test loss**	**Iter.**	**Time**
17	0.87517965	0.78418415	14	4.4946	0.81270048	0.53571040	8	4.1422

Table 5.31: The validation loss, test loss, number of iterations and computation time of the best trial of each method for case 17.

6 | Conclusions and future developments

This work has studied the methodology, which combines Artificial Neural Networks and Wiener-Itô Chaos expansion approximation formulae, first proposed by Funahashi in [26]. According to this paper, applying an ANN directly to the benchmark price (referred as method 1) is inferior to the application of the model to the difference between a benchmark option price and its asymptotic expansion approximation (referred as method 2). This work confirms this result through a deeper investigation.
The methods are tested for several options under different financial models. In particular, the performance of the discussed method is investigated for the first time on a Down-and-In barrier option under CEV and a non linear volatility model.
Moreover, this study gives additional insight on the impact of the hyper-parameters on the two methods performance. The introduction of the validation set allowed to select the hyper-parameters manually, with an educated guess, and automatically, by tuning them. Overall, both methods improve slightly their results with the optimization procedure.
The prediction accuracy is evaluated with MAE and RMSE, two commonly used error measures. The second method is more accurate for any case examined, producing lower values of MAE and RMSE.
Method 2 offers more stable predictions than method 1. This occurs especially when the underlying evolves according to Heston, CEV and the non linear volatility model: in method 1 predictions are over-estimated or under-estimated in some price ranges, while in method 2 this phenomenon does not occur.

Future studies might investigate method 2 in multiple directions.
First, the robustness with respect to hyper-parameters might be studied more in depth. The comparison of only one tuned and one un-tuned model is not sufficient to state that method 2 shows good accuracy independently on the hyper-parameter configuration. A k-fold cross validation might be performed to have additional insight on this aspect.
Another interesting direction is to examine some variations of the method. For exam-

ple, a question to answer is whether the employment of a higher order Wiener-Itô Chaos expansion approximation formula offers more accurate and stable results. Alternatively, other machine learning models might be tested. Some suggestions are Support Vector Regression, Random Forest, XGBoost, Light Gradient Boosting Machine and Genetic Algorithms, which have already been tested in the option pricing field [23].
Another direction to explore is the application of the method on different options and financial models. Examples to explore are American, Asian and double barrier options, whose approximation closed formulae have been derived in [65] and [25, 58] and [29], respectively. Otherwise, the method might be adopted for fractional volatility models, whose asymptotic expansion of the underlying is discussed in [27].

Bibliography

[1] Ray tune documentation. URL `https://docs.ray.io/en/latest/tune/index.html`.

[2] *Hutter, F., Kotthoff, L., Vanschoren, J. (eds) Automated Machine Learning*, chapter Hyperparameters Optimazation. The Springer Series on Challenges in Machine Learning. Springer, 2019. doi: https://doi.org/10.1007/978-3-030-05318-5_1.

[3] I. A. Abramowitz, Milton Stegun. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*, page 775. National Bureau of Standards Applied Mathematics Series, 10 edition, 12 1972.

[4] H. Albrecher, P. A. Mayer, W. Schoutens, and J. Tistaert. The little heston trap. *Wilmott*, pages 83–92, 2006.

[5] L. Andersen. *Efficient Simulation of the Heston Stochastic Volatility Model*. SSRN, 2007. URL `https://dx.doi.org/10.2139/ssrn.946405`.

[6] D. Aumiller. Mini batch gradient descent, adam and epochs. URL `https://stackoverflow.com/questions/51373903`.

[7] L. Bachelier. Théorie de la spéculation. *Annales scientifiques de l'École Normale Supérieure*, 3e série, 17:21–86, 1900. doi: 10.24033/asens.476. URL `http://www.numdam.org/articles/10.24033/asens.476/`.

[8] P. Baheti. Activation functions in neural networks [12 types use cases]. URL `https://www.v7labs.com/blog/neural-networks-activation-functions`.

[9] P. Baldi. Exact asymptotics for the probability of exit from a domain and applications to simulations. *The Annals of Probability*, 23(4):1644–1670, 1995.

[10] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[11] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparam-

eter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[12] T. Björk. *Arbitrage theory in continuous time*, page 246. Oxford university press, 2020.

[13] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. ISSN 00223808, 1537534X. URL `http://www.jstor.org/stable/1831029`.

[14] Brilliant.org. Artificial neural network, . URL `https://brilliant.org/wiki/artificial-neural-network/`. Retrieved February 2023.

[15] Brilliant.org. Backpropagation, . URL `https://brilliant.org/wiki/backpropagation/`. Retrieved February 2023.

[16] P. Carr, K. Ellis, and V. Gupta. Static hedging of exotic options. *The Journal of Finance*, 53(3):1165–1190, 1998. doi: https://doi.org/10.1111/0022-1082.00048. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/0022-1082.00048`.

[17] V. Chugh. Tuning adam optimizer parameters in pytorch. URL `https://www.kdnuggets.com/2022/12/tuning-adam-optimizer-parameters-pytorch.html`.

[18] M. Claesen and B. D. Moor. Hyperparameter search in machine learning. *CoRR*, abs/1502.02127, 2015. URL `http://arxiv.org/abs/1502.02127`.

[19] J. Cox. Notes on option pricing i: Constant elasticity of variance diffusions. *Unpublished note, Stanford University, Graduate School of Business*, 1975.

[20] G. Di Nunno, B. Øksendal, and F. Proske. *Malliavin Calculus for Lévy Processes with Applications to Finance.* 01 2008. ISBN 978-3540785712. doi: 10.1007/978-3-540-78572-9.

[21] L. Di Santo. Formula approssimanta per la valutazione di opzioni a barriera singola per modelli a volatilità locale. Master's thesis, Politecnico di Milano, 2017.

[22] E. Dral. Learning from machine learning mistakes. URL `https://towardsdatascience.com/2cf3e91306b9`.

[23] I. Florin. Option pricing using machine learning. *Expert Systems with Applications*, 163:113799, 07 2020. doi: 10.1016/j.eswa.2020.113799.

[24] J. Fouque, G. Papanicolaou, R. Sircar, and K. Solna. Singular perturbations in option pricing. *SIAM Journal on Applied Mathematics*, 63(5):1648–1665, June 2003. ISSN 0036-1399. doi: 10.1137/S0036139902401550.

[25] H. Funahashi. A chaos expansion approach under hybrid volatility models. *Quantitative Finance*, 14:1923–1936, 11 2014. doi: 10.1080/14697688.2013.872283.

[26] H. Funahashi. Artificial neural network for option pricing with and without asymptotic correction. *Quantitative Finance*, 21(4):575–592, 2020. URL `https://doi.org/10.1080/14697688.2020.1812702`.

[27] H. Funahashi. Replication scheme for the pricing of european options. *International Journal of Theoretical and Applied Finance*, 24(3), 2021. URL `https://doi.org/10.1142/S021902492150014X`.

[28] H. Funahashi. Sabr equipped with ai wings. *Quantitative Finance*, 23(2):229–249, 2023. URL `https://EconPapers.repec.org/RePEc:taf:quantf:v:23:y:2023:i:2:p:229-249`.

[29] H. Funahashi and T. Higuchi. An analytical approximation for single barrier options under stochastic volatility models. *Annals of Operations Research*, 266:129–157, 2018. URL `https://doi.org/10.1007/s10479-017-2559-3`.

[30] H. Funahashi and M. Kijima. A chaos expansion approach for the pricing of contingent claims. *Research Paper Series*, 99, 2011. URL `https://doi.org/10.21314/JCF.2015.299`.

[31] H. Funahashi and M. Kijima. A chaos expansion approach for the pricing of contingent claims. 2012.

[32] H. Funahashi and M. Kijima. A chaos expansion approach for the pricing of contingent claims. *Journal of Computational Finance*, 18(3):25–58, 2015. URL `https://doi.org/10.21314/JCF.2015.299`.

[33] P. Glasserman. *Monte Carlo Methods in Financial Engineering*, pages 368–370. Springer, New York, NY, USA, 2004.

[34] P. S. Hagan, D. Kumar, A. Lesniewski, and D. E. Woodward. Managing smile risk. *Wilmott Magazine*, September:84–108, 2002.

[35] A. Hernandez. Model calibration with neural networks. *Neuroeconomics eJournal*, 2016.

[36] S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2):327–43, 1993. URL `https://doi.org/10.1093/rfs/6.2.327`.

[37] T. Hodson. Root-mean-square error (rmse) or mean absolute error (mae): when to

use them or not. *Geoscientific Model Development*, 15:5481–5487, 07 2022. URL https://doi.org/10.5194/gmd-15-5481-2022.

[38] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. URL https://doi.org/10.1016/0893-6080(90)90005-6.

[39] B. Horvath, A. Muguruza, and M. Tomas. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance*, 21(1):11–27, 2020. URL https://doi.org/10.1080/14697688.2020.1817974.

[40] J. Hull and A. White. The pricing of options on assets with stochastic volatilities. *The Journal of Finance*, 42(2):281–300, 1987. doi: https://doi.org/10.1111/j.1540-6261.1987.tb02568.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1987.tb02568.x.

[41] A. Itkin. Deep learning calibration of option pricing models: some pitfalls and solutions, 2019. URL https://arxiv.org/abs/1906.03507.

[42] K. Itô. Multiple wiener integral. *Journal of the Mathematical Society of Japan*, 3(1):157–169, 1951. URL https://doi.org/10.2969/jmsj/00310157.

[43] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

[44] L. Li, K. Jamieson, A. Rostamizadeh, K. Gonina, M. Hardt, B. Recht, and A. Talwalkar. Massively parallel hyperparameter tuning, 2018. URL https://openreview.net/forum?id=S1Y7OOlRZ.

[45] P. Liashchynskyi and P. Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for NAS. *CoRR*, abs/1912.06059, 2019. URL http://arxiv.org/abs/1912.06059.

[46] S. Liu, A. Borovykh, L. A. Grzelak, and C. W. Oosterlee. A neural network-based framework for financial model calibration. *Journal of Mathematics in Industry*, 9(1), sep 2019. doi: 10.1186/s13362-019-0066-7.

[47] J. D. Macbeth and L. J. Merville. Tests of the black-scholes and cox call option valuation models. *The Journal of Finance*, 35(2):285–301, 1980. doi: https://doi.org/10.1111/j.1540-6261.1980.tb02157.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1980.tb02157.x.

[48] D. Mack. How to pick the best learning rate for your machine learning project. URL `https://medium.com/octavian-ai/5acb418f9b2`.

[49] D. Marris. Financial option pricing and skewed volatility. Master's thesis, University of Cambridge, 1999. MPhil Statistical Science. External Supervisor: Dr Martin Baxter Nomura International Plc.

[50] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 12 1943. doi: https://doi.org/10.1007/BF02478259.

[51] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models, 2017.

[52] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 6 1958. doi: https://doi.org/10.1037/h0042519.

[53] M. Rubinstein and E. Reiner. Breaking down the barriers. *RISK*, 4:28–35, 1991.

[54] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 10 1986. doi: https://doi.org/10.1038/323533a0.

[55] D. Sculley, J. Snoek, A. B. Wiltschko, and A. Rahimi. Winner's curse? on pace, progress, and empirical rigor. In *International Conference on Learning Representations*, 2018.

[56] Y. Shevchuk. What is batch size in neural network? URL `https://stats.stackexchange.com/questions/153531`.

[57] K. Shiraya. An approximation method for pricing continuous barrier options under multi-asset local stochastic volatility models. *International Journal of Theoretical and Applied Finance*, 23, 11 2020. doi: 10.1142/S021902492050051X.

[58] K. Shiraya and A. Takahashi. Pricing average options on commodity. *Journal of Futures Markets*, 31:407 – 439, 05 2011. doi: 10.1002/fut.20481.

[59] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018. URL `https://arxiv.org/abs/1803.09820`.

[60] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[61] J. D. Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18(10):1635–1643, October 2018. doi: 10.1080/14697688.2018.149. URL `https://ideas.repec.org/a/taf/quantf/v18y2018i10p1635-1643.html`.

[62] A. Takahashi. An asymptotic expansion approach to pricing financial contingent claims. *Asia-Pacific Financial Markets*, 6:115–151, 1999. URL `https://doi.org/10.1023/A:1010080610650`.

[63] A. Takahashi and K. Takehara. An asymptotic expansion approach to currency options with a market model of interest rates under stochastic volatility processes of spot exchange rates. *Asia-Pacific Financial Markets*, 14(1):69–121, 2007. URL `https://EconPapers.repec.org/RePEc:kap:apfinm:v:14:y:2007:i:1:p:69-121`.

[64] N. Yoshida. Asymptotic expansions for statistics related to small diffusions. *Journal of Japan Statistical Society*, 22:139–159, 1992.

[65] S. M. Zhang and Y. Feng. American option pricing under the double Heston model based on asymptotic expansion. *Quantitative Finance*, 19(2):211–226, February 2019. doi: 10.1080/14697688.2018.147. URL `https://ideas.repec.org/a/taf/quantf/v19y2019i2p211-226.html`.

[66] B. Øksendal. *Stochastic Differential Equations*, chapter 5, pages 65–84. Universitext. Springer, Berlin, Heidelberg, 6 edition, 2003.

[67] B. Øksendal. *Stochastic Differential Equations*. Universitext. Springer, Berlin, Heidelberg, 6 edition, 2003. ISBN 978-3-540-04758-2. doi: https://doi.org/10.1007/978-3-642-14394-6.

A | Appendix A

This appendix shows the exact formulae needed to compute the price of a European Call option. They hold in the most general case, under the Local Stochastic Volatility Model (LSVM). Funahashi derived them in [25].

Let $\sigma^{(0)}(t) = \sigma(F(0,t), V(0,t))$ and $\gamma^{(0)}(t) = \gamma(V(0,t))$ where $F(0,t) = S_0 e^{\int_0^t r(s)\,ds}$ and $V(0,t) = \overline{E}(t)(\nu_0 + \int_0^t E(u)\theta(u)\,du)$ with $E(t) = e^{\int_0^t \kappa(u)\,du}$ and $\overline{E}(t) = \frac{1}{E(t)}$. Then $p_i(t)$ are defined as:

$$\begin{aligned}
p_1(s) &\coloneqq \sigma^{(0)}(s) + \left(\sigma_S^{(0)}(s)F(0,s) + \frac{1}{2}F(0,s)^2\sigma_{SS}^{(0)}(s)\right)\left(\int_0^s (\sigma_0(u))^2\,du\right) \\
&\quad + \frac{1}{2}\sigma_{\nu\nu}^{(0)}(s)\overline{E}(s)^2\left(\int_0^s E(u)^2\gamma^{(0)}(u)^2\,du\right) \\
&\quad + \left(\sigma_\nu^{(0)}(s)\overline{E}(s) + \sigma_{S\nu}^{(0)}(s)F(0,s)\overline{E}(s)\right)\left(\int_0^s \rho E(u)\gamma^{(0)}(u)\sigma^{(0)}(u)\,du\right), \\
p_2(s) &\coloneqq \sigma^{(0)}(s) + F(0,s)\sigma_S^{(0)}(s), \\
p_3(s) &\coloneqq \sigma_\nu^{(0)}\overline{E}(s), \\
p_4(s) &\coloneqq E(s)\gamma^{(0)}(s), \\
p_5(s) &\coloneqq \sigma^{(0)}(s) + 3\sigma_S^{(0)}(s)F(0,s) + \sigma_{SS}^{(0)}F(0,s)^2, \\
p_6(s) &\coloneqq \sigma_{\nu\nu}^{(0)}\overline{E}^2, \\
p_7(s) &\coloneqq \sigma_\nu^{(0)}\overline{E}(s) + \sigma_{S\nu}^{(0)}F(0,s)\overline{E}(s), \\
p_8(s) &\coloneqq \sigma_S^{(0)}(s)F(0,s).
\end{aligned}$$

Given $S_t^{(0)} = F(0,t)$ and $\nu_t^{(0)} = V(0,t)$, the derivatives are defined as:

$$\begin{aligned}
\sigma_S^{(0)}(t) &\coloneqq \partial_S\sigma(s,\nu)\big|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}, & \sigma_\nu^{(0)}(t) &\coloneqq \partial_\nu\sigma(s,\nu)\big|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}, \\
\sigma_{SS}^{(0)}(t) &\coloneqq \partial_{SS}\sigma(s,\nu)\big|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}, & \sigma_{\nu\nu}^{(0)}(t) &\coloneqq \partial_{\nu\nu}\sigma(s,\nu)\big|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}, \\
\sigma_{S\nu}^{(0)}(t) &\coloneqq \partial_{S\nu}\sigma(s,\nu)\big|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}.
\end{aligned}$$

Σ_t, $q_i(t)$, $q_{2,k}(t)$ and $q_{4,k}(t)$ are defined as:

$$
\begin{aligned}
\Sigma_t &\coloneqq \int_0^t p_1(s)^2\,ds,\\
q_1(t) &\coloneqq \int_0^t p_1(s)p_2(s)\left(\int_0^s \sigma^{(0)}(u)p_1(u)\,du\right)ds + \int_0^t p_1(s)p_3(s)\left(\int_0^s \rho p_1(u)p_4(u)\,du\right)ds,\\
q_2(t) &\coloneqq q_{2,1}(t)+q_{2,2}(t)+q_{2,3}(t),\\
q_3(t) &\coloneqq q_1(t)^2,\\
q_4(t) &\coloneqq q_{4,1}(t)+q_{4,2}(t)+q_{4,3}(t),\\
q_5(t) &\coloneqq \int_0^t p_2(s)^2\left(\int_0^s (\sigma^{(0)}(u))^2\,du\right)ds + \int_0^t p_3(s)^2\left(\int_0^s p_4(u)^2\,du\right)ds\\
&\quad + 2\int_0^t p_2(s)p_3(s)\left(\int_0^s \rho\sigma^{(0)}(u)p_4(u)\,du\right)ds,\\
q_{2,1}(t) &\coloneqq \int_0^t p_1(s)p_5(s)\left(\int_0^s \sigma^{(0)}(u)p_1(u)\left(\int_0^u \sigma^{(0)}(v)p_1(v)\,dv\right)du\right)ds\\
&\quad + \int_0^t p_1(s)p_2(s)\left(\int_0^s p_1(u)p_8(u)\left(\int_0^u \sigma^{(0)}(v)p_1(v)\,dv\right)du\right)ds,\\
q_{2,2}(t) &\coloneqq \int_0^t p_1(s)p_3(s)\left(\int_0^s \rho p_1(u)\gamma_\nu^{(0)}(u)\left(\int_0^u \rho p_1(v)p_4(v)\,dv\right)du\right)ds,\\
q_{2,3}(t) &\coloneqq \int_0^t p_1(s)p_6(s)\left(\int_0^s \rho p_1(u)p_4(u)\left(\int_0^u \rho p_1(v)p_4(v)\,dv\right)du\right)ds\\
&\quad + \int_0^t p_1(s)p_7(s)\left(\int_0^s \sigma^{(0)}(u)p_1(u)\left(\int_0^u \rho p_1(v)p_4(v)\,dv\right)du\right)ds\\
&\quad + \int_0^t p_1(s)p_2(s)\left(\int_0^s \rho p_1(u)p_3(u)\left(\int_0^u \rho p_1(v)p_4(v)\,dv\right)du\right)ds,\\
q_{4,1}(t) &\coloneqq 2\int_0^t p_1(s)p_2(s)\left(\int_0^s p_1(u)p_2(u)\left(\int_0^v (\sigma^{(0)}(v))^2\,dv\right)du\right)ds\\
&\quad + 2\int_0^t p_1(s)p_2(s)\left(\int_0^s \sigma^{(0)}(u)p_2(u)\left(\int_0^v \sigma^{(0)}(v)p_1(v)\,dv\right)du\right)ds\\
&\quad + \int_0^t p_2(s)^2\left(\int_0^s \sigma^{(0)}(u)p_1(u)\,du\right)^2 ds,\\
q_{4,2}(t) &\coloneqq 2\int_0^t p_1(s)p_3(s)\left(\int_0^s p_1(u)p_3(u)\left(\int_0^v p_4(v)^2\,dv\right)du\right)ds\\
&\quad + 2\int_0^t p_1(s)p_3(s)\left(\int_0^s \rho p_3(u)p_4(u)\left(\int_0^u \rho p_1(v)p_4(v)\,dv\right)du\right)ds\\
&\quad + \int_0^t p_3(s)^2\left(\int_0^s \rho p_1(u)p_4(u)\,du\right)^2 ds,
\end{aligned}
$$

$$\begin{aligned}
q_{4,3}(t) \coloneqq\, & 2\int_0^t p_1(s)p_2(s)\left(\int_0^s p_1(u)p_3(u)\left(\int_0^v \rho\sigma^{(0)}(v)p_4(v)\,dv\right)\,du\right)\,ds \\
& + 2\int_0^t p_1(s)p_3(s)\left(\int_0^s p_1(u)p_2(u)\left(\int_0^v \rho\sigma^{(0)}(v)p_4(v)\,dv\right)\,du\right)\,ds \\
& + 2\int_0^t p_1(s)p_2(s)\left(\int_0^s \sigma^{(0)}(u)p_3(u)\left(\int_0^v \rho p_1(v)p_4(v)\,dv\right)\,du\right)\,ds \\
& + 2\int_0^t p_1(s)p_3(s)\left(\int_0^s \rho p_2(u)p_4(u)\left(\int_0^v \sigma^{(0)}(v)p_1(v)\,dv\right)\,du\right)\,ds \\
& + 2\int_0^t p_2(s)p_3(s)\left(\int_0^s \sigma^{(0)}(u)p_1(u)\left(\int_0^u \rho p_1(v)p_4(v)\,dv\right)\,du\right)\,ds.
\end{aligned}$$

B | Appendix B

This appendix shows the exact formulae needed to compute the price of Barrier options. They hold in the most general case, under the Local Stochastic Volatility Model (LSVM). Funahashi derived them in [29].

Let $\sigma^{(0)}(t) = \sigma(F(0,t), V(0,t))$ and $\gamma^{(0)}(t) = \gamma(V(0,t))$ where $F(0,t) = S_0 e^{\int_0^t r(s)\,ds}$ and $V(0,t) = \overline{E}(t)(\nu_0 + \int_0^t E(u)\theta(u)\,du)$ with $E(t) = e^{\int_0^t \kappa(u)\,du}$ and $\overline{E}(t) = \frac{1}{E(t)}$. Moreover, the derivatives of $\sigma^{(0)}(t)$ are $\sigma_S^{(0)}(t) := \partial_S \sigma(s,\nu)|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}$, $\sigma_\nu^{(0)}(t) := \partial_\nu \sigma(s,\nu)|_{s=S_t^{(0)},\nu=\nu_t^{(0)}}$. Let $\Sigma_t := \int_0^t (\sigma^{(0)}(s))^2\,ds$, and

$$q(t) := \int_0^t p_1(s)\sigma^{(0)}(s)\left(\int_0^s (\sigma^{(0)}(u))^2\,du\right)ds + \rho\int_0^t p_2(s)\sigma^{(0)}(s)\left(\int_0^s p_3(u)\sigma^{(0)}(u)\,du\right)ds,$$

where

$$p_1(s) := \sigma^{(0)}(s) + F(0,s)\sigma_S^{(0)}(s),$$
$$p_2(s) := \sigma_\nu^{(0)}\overline{E}(s),$$
$$p_3(s) := E(s)\gamma^{(0)}(s).$$

Then $\omega_t^1(B)$, Ω_T, $\dot{\omega}_T$,$\ddot{\omega}_T$, ϕ, $X_i(T)$ and $f_i(T)$ are defined as:

$$\omega_t^1(B) := -\frac{\sqrt{t}(\Sigma_t^{\frac{3}{2}} - \sqrt{D_t(B)})}{2q(t)},$$

where

$$D_t(B) := \Sigma_t^3 + 4\left(\frac{B}{F(0,t)} - 1\right)q(t)\Sigma_t + 4q^2(t),$$

$$\Omega_t := \int_0^t \alpha^2(s)\,ds,$$

where

$$\alpha(t) := -\frac{\partial}{\partial t}\omega_t^1(B),$$

$$\dot{\omega}_T := \omega_T^1(B) + \omega_0^1(B),$$

$$\ddot{\omega}_T \coloneqq \dot{\omega}_T - \omega_T^1(K),$$

$$\phi \coloneqq \frac{\int_0^t \alpha(s)\,ds}{|\int_0^t \alpha(s)\,ds|},$$

$$\begin{aligned}
X_1(T) \coloneqq{}& F(0,T)q(T)\left(\omega_T^1(K)+\dot{\omega}_T\right)\left[(\omega_T^1(K))^2+(\dot{\omega}_T)^2\right]\Omega_T T^{-\frac{3}{2}}\\
&+F(0,T)\left[(\omega_T^1(K))^2+\omega_T^1(K)\dot{\omega}_T+(\dot{\omega}_T)^2\right]\left(2\phi q(T)\sqrt{\Omega_T}+\Sigma_T^{\frac{3}{2}}\Omega_T\right)T^{-1}\\
&+\left(\omega_T^1(K)+\dot{\omega}_T\right)\Sigma_T\left[F(0,T)\left(2\phi\sqrt{\Sigma_T\Omega_T}+\Omega_T\right)-K\Omega_T\right]T^{-\frac{1}{2}}\\
&+F(0,T)q(T)\left[\omega_T^1(K)(2+\Omega_T)+\dot{\omega}_T(2+3\Omega_T)\right]T^{-\frac{1}{2}}\\
&+F(0,T)\Sigma_T\left[2\phi\sqrt{\Omega_T}+\sqrt{\Sigma_T}(2+\Omega_T)\right]\\
&+2\phi(F(0,T)q(T)-K\Sigma_T)\sqrt{\Omega_T},\\
X_2(T) \coloneqq{}& F(0,T)q(T)\left(\omega_T^1(B)+\dot{\omega}_T\right)\left[(\omega_T^1(B))^2+(\dot{\omega}_T)^2\right]\Omega_T T^{-\frac{3}{2}}\\
&+F(0,T)\left[(\omega_T^1(B))^2+\omega_T^1(B)\dot{\omega}_T+(\dot{\omega}_T)^2\right]\left(2\phi q(T)\sqrt{\Omega_T}+\Sigma_T^{\frac{3}{2}}\Omega_T\right)T^{-1}\\
&+\left(\omega_T^1(B)+\dot{\omega}_T\right)\Sigma_T\left[F(0,T)\left(2\phi\sqrt{\Sigma_T\Omega_T}+\Omega_T\right)-K\Omega_T\right]T^{-\frac{1}{2}}\\
&+F(0,T)q(T)\left[\omega_T^1(B)(2+\Omega_T)+\dot{\omega}_T(2+3\Omega_T)\right]T^{-\frac{1}{2}}\\
&+F(0,T)\Sigma_T\left[2\phi\sqrt{\Omega_T}+\sqrt{\Sigma_T}(2+\Omega_T)\right]\\
&+2\phi(F(0,T)q(T)-K\Sigma_T)\sqrt{\Omega_T},\\
X_3(T) \coloneqq{}& F(0,T)q(T)\dot{\omega}_T^4\Omega_T T^{-2}\\
&+F(0,T)\dot{\omega}_T^3\left(2\phi q(T)\sqrt{\Omega_T}+\Sigma_T^{\frac{3}{2}}\Omega_T\right)T^{-\frac{3}{2}}\\
&+\dot{\omega}_T^2\left(2\phi F(0,T)\Sigma_T\sqrt{\Sigma_T\Omega_T}+F(0,T)\Sigma_T\Omega_T+2F(0,T)q(T)(1+2\Omega_T)-K\Sigma_T\Omega_T\right)T^{-1}\\
&+2F(0,T)\dot{\omega}_T\left(2\phi q(T)\sqrt{\Omega_T}+\Sigma_T^{\frac{3}{2}}(1+\Omega_T)+\phi\Sigma_T\sqrt{\Omega_T}\right)T^{-\frac{1}{2}}\\
&-2\phi K\dot{\omega}_T\sqrt{\Omega_T}\Sigma_T T^{-\frac{1}{2}}\\
&+2F(0,T)\Sigma_T\left(1+\phi\sqrt{\Sigma_T\Omega_T}\right)+F(0,T)q(T)\Omega_T-2K\Sigma_T,\\
f_1(T) \coloneqq{}& \sqrt{\frac{2T}{\pi}}e^{-\frac{\ddot{\omega}_T^2}{2T}}\Big\{-KT\Sigma_T\sqrt{\Omega_T}\left(2\phi\sqrt{T}+\sqrt{\Omega_T}(\dot{\omega}_T+\omega_T^1(K))\right)\\
&+F(0,T)\Big[T^{\frac{3}{2}}\left(2\phi q(T)\sqrt{\Omega_T}+2\phi\Sigma_T\sqrt{\Omega_T}+\Sigma_T^{\frac{3}{2}}(2+\Omega_T)\right)\\
&+\sqrt{T}\left(2\phi q(T)+\Sigma_T^{\frac{3}{2}}\sqrt{\Omega_T}\right)\sqrt{\Omega_T}(\dot{\omega}_T^2+\dot{\omega}_T\omega_T^1(K)+(\omega_T^1(K))^2)\\
&+T\Big(\Sigma_T\left(2\phi\sqrt{\Omega_T\Sigma_T}+\Omega_T\right)(\dot{\omega}_T+\omega_T^1(K))\\
&+q(T)(2+3\Omega_T)\dot{\omega}_T+(2+\Omega_T)\omega_T^1(K))\Big)\\
&+q(T)\Omega_T(\ddot{\omega}_T^3+4\omega_T^1(K)\dot{\omega}_T^2-2(\omega_T^1(K))^2\ddot{\omega}_T)\Big]\Big\},
\end{aligned}$$

$$
\begin{aligned}
f_2(T) \coloneqq {} & -KT\Sigma_T(2T + 2\phi\sqrt{T\Omega_T}\dot{\omega}_T + \Omega_T\dot{\omega}_T^2) \\
& + F(0,T)\Big[2T^2(\Sigma_T + \phi\Sigma_T^{\frac{3}{2}}\sqrt{\Omega_T} + q(T)\Omega_T) \\
& + 2T^{\frac{3}{2}}\left(2\phi q(T)\sqrt{\Omega_T} + \phi\Sigma_T\sqrt{\Omega_T} + \Sigma_T^{\frac{3}{2}}(1+\Omega_T)\right)\dot{\omega}_T + q(T)\Omega_T\dot{\omega}_T^4 \\
& + T(2\phi\Sigma_T^{\frac{3}{2}}\sqrt{\Omega_T} + \Sigma_T\Omega_T + q(T)(2+4\Omega_T))\dot{\omega}_T^2 + \sqrt{T\Omega_T}(2\phi q(T) + \Sigma_T^{\frac{3}{2}}\Omega_T)\dot{\omega}_T^3\Big].
\end{aligned}
$$

List of Figures

List of Tables

List of Symbols

Variable	**Description**
ANN	Artificial Neural Network
B&S	Black & Scholes model
HES	Heston model
CEV	Constant Elasticity of Variance model
NLV	Non linear volatility model
MC	Monte Carlo
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
ATM	At-the-money

Acknowledgements

Questo percorso al Politecnico di Milano che ho appena concluso sarebbe stato molto più difficile se non avessi avuto attorno a me le persone che mi hanno sempre sostenuta. Non posso dimenticare i tanti ostacoli che si sono presentati durante questi anni universitari e, di conseguenza, l'aiuto fondamentale di parenti e amici per affrontare i momenti di maggiore sconforto. Per questo motivo è molto importante per me condividere la soddisfazione di questo traguardo con le persone che più amo.
Grazie ai miei genitori, Annalisa e Raffaello, e mio fratello Leonardo che sono stati partecipi di tutti i miei sacrifici e mi hanno sostenuta sempre attraverso il loro amore incondizionato.
Grazie a tutti i miei parenti. In particolare, devo un grazie immenso ai miei nonni: Gino, per avermi motivata negli studi sin da bambina; Dino, per avermi conferito il titolo di "scienziata"; Adriana, per avermi trasmesso la passione per la matematica; Lucia, per esser stata il mio più grande esempio di dedizione e meticolosità.
Grazie ai miei amici del liceo, soprattutto alle mie "raccattate", Elena, Caterina, Martina ed Elisa, che sono il mio punto di riferimento per dare ascolto e ridimensionare tutte le mie preoccupazioni da più di dieci anni.
Grazie ai miei amici "del giornalino", o "giornalisti". Anche se per qualcuno di loro questo appellativo può risultare improprio, sanno bene che mi riferisco a Samantha, Riccardo, Elena, gli Andrea, Paolo, Francesco e Marco. Mi hanno offerto tantissime occasioni per distrarmi dagli studi e mi hanno permesso di credere di più in me stessa, dimostrando tanta stima nei miei confronti, che è assolutamente reciproca.
Grazie ai miei coinquilini di Milano, in particolare alle fantastiche ragazze che hanno abitato via Calzecchi: Anna, Marta, Stefania, Roberta, Carolina e Samira. Dal primo giorno mi hanno guidata nel muovere i miei primi passi a Milano e mi hanno fatta sentire a casa. Quella quotidianità non ritornerà mai più, ma custodirò sempre tanti bei ricordi di momenti felici. Sono veramente fortunata che le loro strade si siano incrociate con la mia.
Grazie a Pilar, amica fidata e sincera. La distanza non avrà mai la meglio sulla nostra amicizia.
Grazie ai "finti universitari", con cui ho condiviso tanti episodi divertenti nelle aule del

Politecnico.
Grazie al mio relatore Professor Marazzina per la sua cortese disponibilità.
Merci beaucoup à mes amis français: Margaux, Alexandre, Arthur et Eliott. Vous m'avez accueillie à bras ouvertes et, graçe à vous, j'ai veçu la meilleure expérience ERASMUS que je pouvais imaginer.

Milano, 4 maggio 2023 Valentina